VŠB – Technická univerzita Ostrava

Fakulta elektrotechniky a informatiky

Katedra kybernetiky a biomedicínského inženýrství

Detekce pohybu z video záznamu

Movement Detection from Video Recording

2016 **Tomáš Vilím**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra kybernetiky a biomedicínského inženýrství

Zadání bakalářské práce

Student: **Tomáš Vilím**

Studijní program: B2649 Elektrotechnika

Studijní obor: 3901R039 Biomedicínský technik

Téma: Detekce pohybu z video záznamu
Movement Detection from Video Recording

Jazyk vypracování: čeština

Zásady pro vypracování:

1. Nastudování problematiky zpracování dynamických obrazových dat.
2. Nastudování problematiky detekce pohybu ve video záznamech.
3. Nastudování metod, které se využívají pro detekci objektů ve videozáznamech.
4. Rešerše metod, které se využívají pro analýzu pohybu ve videozáznamech.
5. Návrh a realizace algoritmu pro detekci pohybu z videozáznamu.
6. Vyhodnocení navrženého řešení na reálných obrazových datech.
7. Implementace softwarového prostředí pro analýzu a detekci pohybu z videozáznamu.

Seznam doporučené odborné literatury:

[1] REYES-ALDASORO, Constantino Carlos. *Biomedical image analysis recipes in MATLAB: for life scientists and engineers.* 1 edition. Wiley-Blackwell (June 22, 2015. 416p. ISBN 978-1118657553.
[2] NIXON, Mark S a Alberto S AGUADO. *Feature extraction & image processing for computer vision.* 3rd ed. Oxford: Academic Press, 2012, xvii, 609 p., [2] p. of plates. ISBN 0123965497.
[3] BATCHELOR, Bruce G. (ed.) *Machine vision handbook.* London: Springer-Verlag, c2012. ISBN 978-1-84996-168-4.
[4] MARQUES, Oge. *Practical image and video processing using MATLAB.* Hoboken, N.J.: Wiley-IEEE Press, 2011. 639 p. ISBN-13: 978-0470048153.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Jan Kubíček**

Datum zadání: 01.09.2015

Datum odevzdání: 29.04.2016

doc. Ing. Jiří Koziorek, Ph.D.	prof. RNDr. Václav Snášel, CSc.
vedoucí katedry	*děkan fakulty*

Prohlášení:

Prohlašuji, že svou bakalářskou práci jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Ostravě dne 29. dubna 2016

Tomáš Silva

.....

podpis autora

Abstrakt:

Tato bakalářská práce je zaměřena na návrh a implementaci softwarového prostředí pro analýzu a detekci pohybu z video záznamu. Dále práce obsahuje vysvětlení problematiky zpracování dynamických obrazových dat a problematiku detekce pohybu ve video záznamech, řešené v rešeršní části. Hlavní částí bakalářské práce je návrh algoritmu pro detekci pohybu z video záznamu. Jednotlivé kroky navrhovaného algoritmu jsou v textu podrobně popsány. Algoritmus byl otestován na experimentálně vytvořených video záznamech a výsledky detekce pohybu byly zhodnoceny v závěru.

Klíčová slova:

Detekce pohybu, parametry obrazu, zpracování obrazu, zpracování videa, metoda odečtení pozadí, frame differencing method, binární obraz, monochromatický obraz

Abstract:

This bachelor thesis is devoted to the theme of movement detection from video recording. Next part of the bachelor thesis is dedicated to complex research of methods which can be used for a movement deteciton. The main part of this thesis is dedicated to proposed algorithm for movement detection. Each step of the proposed algorithm is described in details in the following text. The algorithm has been tested on real data and outcome of the detection has been evaluated in conclusion.

Key words:

Movement detection, video parameters, video processing, image processing, background subtraction, frame differencing, binary image, grayscale image

Poděkování:

Rád bych poděkoval svému vedoucímu panu Ing. Janu Kubíčkovi za odborné vedení, cenné rady, věcné připomínky a vstřícnost při konzultacích, které mi pomohly tuto bakalářskou práci zkompletovat.

Obsah

Seznam obrázků

1 Úvod

Cílem této bakalářské práce je vytvoření algoritmu pro detekci pohybu z video záznamu. Obsahem této práce není pouze prezentování navrženého algoritmu, ale také seznámení se základními parametry obrazu a video záznamu, a také s principy použitých metod pro vývoj algoritmu. Tyto oblasti jsou nezbytné pro pochopení, k čemu algoritmus slouží a jak pracuje. První teoretická část bakalářské práce je zaměřená na rozbor digitálního obrazu, princip jeho zpracování, vzorkování a následné uložení v paměti počítače. Shrnuje základní parametry digitálního obrazu jako je jas, rozlišení, bitová hloubka, barevný model atd. Zbytek teoretické části je věnován dynamickému obrazu, procesu jeho ukládání do paměti, kompresi a převzorkování. Do teoretické části je také zahrnuta rešerše současné problematiky detekce retinálních lézí. Zde jsou shrnuty již existující metody detekce pohybu z video záznamu. V praktické části je rozebrán návrh a realizace algoritmu pro detekci pohybu. Navržený algoritmus je dále testován na reálných datech a výsledky jsou srovnány. Součástí je také postup tvorby graficko-uživatelského rozhraní pro detekci pohybu z video záznamů.

2 Digitální obraz

Digitální obraz je obrazová informace, která je uložená v digitální paměti. Pro správné uložení obrazové informace musí dojít nejprve k digitalizaci, kdy je spojitá funkce převedena na diskrétní a všechny spojité hodnoty transformovány na jednotky obrazu (pixel – z angl. picture element). [3, 4, 5]

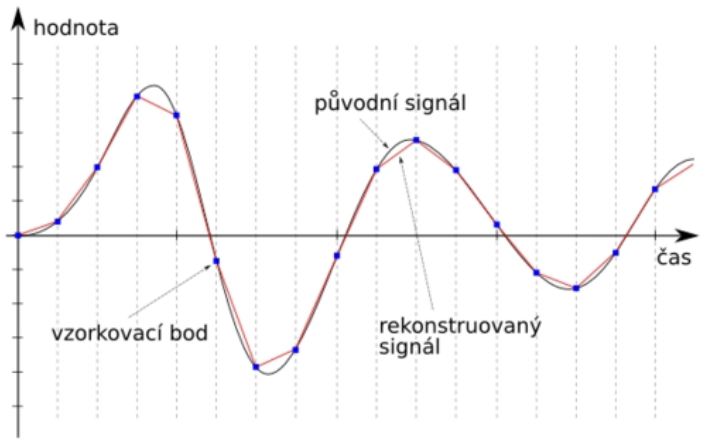
2.1 Digitalizace

Je proces, který se používá při převodu jakéhokoliv analogového signálu (spojitého) na signál digitální (číselný). Výsledný digitální signál je nespojitý a číselný údaj je obvykle kódován v binární soustavě. Během digitalizace vždy dochází ke ztrátě informace, jsme však schopni tuto ztrátu ovlivnit vhodným zvolením vzorkovací frekvence. Hlavní výhoda digitalizace je, že je dobře odolná proti šumu. Výsledný binární signál jsme schopni lépe přenášet, dále s ním manipulovat, ukládat a neomezeně reprodukovat bez ztráty informace. Celý proces digitalizace spočívá ve dvou nezávislých procesech vzorkování a následném kvantování. [6]

2.1.1 Vzorkování

Při vzorkování (sampling) jsou odebírány diskrétní hodnoty, vzorky, ze spojité funkce. Je zvolena vzorkovací frekvence, která určí, s jakou pravidelností budeme vzorky odebírat. Vstupní signál je transformován na posloupnost diskrétních hodnot, reprezentovanou okamžitými hodnotami vzorkovaného signálu. Jelikož analogový signál je nekonečný a lze jeho časovou základnu donekonečna zvětšovat, dochází při vzorkování vždy ke ztrátě, výsledná digitalizovaná informace není nikdy kompletní. Platí zde, že čím větší vzorkovací frekvenci použijeme, tím kvalitnější obraz, co se týče rozlišení, získáme. Při volení vhodné frekvence však musí být dodržen Shannonův teorém: *„Přesná rekonstrukce spojitého, frekvenčně omezeného signálu z jeho vzorků je možná tehdy, pokud byla vzorkovací frekvence vyšší než dvojnásobek nejvyšší harmonické složky vzorkovaného signálu.“* [3, 9]

$$f_v > 2f_{MAX\,[s^{-1}]} \tag{1}$$

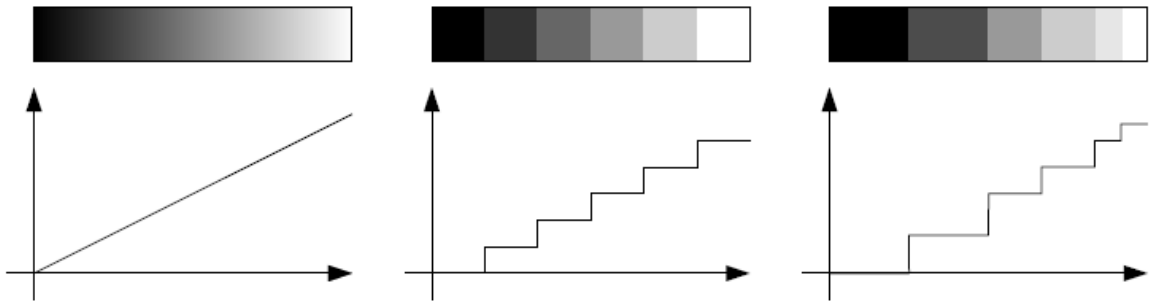


Obrázek 1 - Ukázka vzorkování analogového signálu [9]

2.1.2 Kvantování

Při kvantování dochází k převodu spojité hodnoty, např. jasu na celočíselný rozsah. Podle počtu bitů, které máme k dispozici, můžeme rozlišit různé kvantizační úrovně. Nejpoužívanější je 8bit paměť, kde jsme schopni rozlišit 0-255 úrovní jasu. Stejně jako u vzorkování, tak i u kvantizace, dochází ke kvantizační chybě. Chyby jsme schopni eliminovat, zvýšením počtu kvantizačních úrovní, což má za následek zvětšení bitové paměti a větší výpočetní náročnost. [3, 4]

Digitální obraz je v paměti počítače reprezentován jako matice prvků, pixelů. Jejich počet je konečný a závisí na rozlišení výsledného digitálního obrazu. Každému pixelu je přiřazena jedna hodnota jasu, podle hloubky bitové paměti a kvality snímku. [5]



Obrázek 2 - Ukázka kvantovnání obrazu [9]

2.2 Parametry digitálního obrazu

Za parametry obrazu lze považovat souhrn vlastností, které daný obrazový záznam reprezentují (jas, rozlišení, bitová hloubka, barevný model, atd.). Velkou výhodou je, že pomocí různých softwarových programů pro úpravu obrazu jsme schopni tyto parametry upravovat na požadovanou úroveň. [5, 6]

2.2.1 Jas

Jas obrazu je reprezentován rozložením jasových úrovní v obraze, a také je dán parametry dané jasové stupnice. To vše je určeno již při samotném procesu kvantování. Kolik pixelů obrazu reprezentuje konkrétní jasovou hladinu, vyjadřuje histogram. V souvislosti s jasem můžeme použít různé jasové transformace (transformace bodové, transformace jasové stupnice). [7]

2.2.2 Rozlišení obrazu

Rozlišení obrazu reprezentuje hustotu obrazové informace, vyjadřuje jemnost pixelové reprezentace obrazu. Při digitalizaci udává s jakou jemností je rozlišována předloha z hlediska vzorkovací frekvence a kvantizace. Při zobrazování je rozlišení dáno počtem pixelů, který výsledný obraz reprezentují. Čím je rozlišení větší, tím větší počet pixelů tvoří daný obraz, při stejném rozměru plochy. Rozlišení můžeme udávat dvěma způsoby relativně a absolutně. Relativně se rozlišení udává v bodech nebo pixelech na jednotku délky (dpi[1], ppi[2], ppc[3]). Pro určení absolutního rozlišení musíme znát velikost plochy, která je rovna násobku pixelů na ose X a Y. [7]

[1] Dot per Inch (Počet bodů na 25,4 mm)
[2] Pixel per Inch (zobrazeno pixelů na 25,4 mm)
[3] Pixel per Centimeter (zobrazeno pixelů na 1 cm)

Zde je příklad několika standardizovaných rozlišení: [7]

- VGA (Video Graphic Array – 1987) 640×480 = 307 200 px 4:3
- HD TV 720p – 1280×720 (High-definition television – HD ready progressive) 16:9
- WQXGA (Wide Quad eXtended Graphics Array) 2560×1600 = 4,1 Mpx 16:10

2.2.2.1 Interpolace

S rozlišením obrazu úzce souvisí transformace rozlišení. Jednou z těchto změn rozlišení je interpolace, metoda, při níž se výpočtem uměle zvětšuje nebo snižuje počet obrazových bodů. Interpolaci obrazu nejčastěji provádíme při změně rozlišení, změně velikosti, změně počtu použitých barev. [3]

Algoritmicky nejjednodušší interpolací je metoda nejbližšího souseda (nearest neighbor), kdy nový pixel přebírá vlastnosti předchozího, tato metoda však nepřináší nejlepší výsledky, co se výsledné kvality obrazu týká. Výhodou je však nízká výpočetní náročnost. [4]

Další metodou interpolace je interpolace lineární, kdy se nová hodnota vypočte jako vážený průměr dvou sousedních hodnot. U dvojrozměrného obrazu je tedy nutné ji provést dvakrát pro řádek a dvakrát pro sloupec. Nejlepší formou lineární interpolace je metoda bikubická, kdy se nová hodnota vypočte ze čtyř sousedních bodů. [5]

V souvislosti s interpolací můžeme zavést dva druhy interpolačních algoritmů z hlediska adaptivity. Neadaptivní algoritmy provedou stejnou úpravu pro celý obraz, oproti tomu algoritmy adaptivní aplikují rozdílnou metodu interpolace na základě rozdílných ploch (například jiný algoritmus na hrany, než na zbarvené plochy). [6]

2.2.3 Geometrické transformace obrazu

Rozumíme takové transformace obrazu, které mají za následek geometrické změny (změna rozlišení, zvětšení, zmenšení, zkosení, posunutí, otáčení). Při těchto geometrických změnách vzniká problém nespojitosti obrazu (vznik děr, vznik velkého počtu pixelů). Tento problém můžeme vyřešit převzorkováním s následnou interpolací. [9]

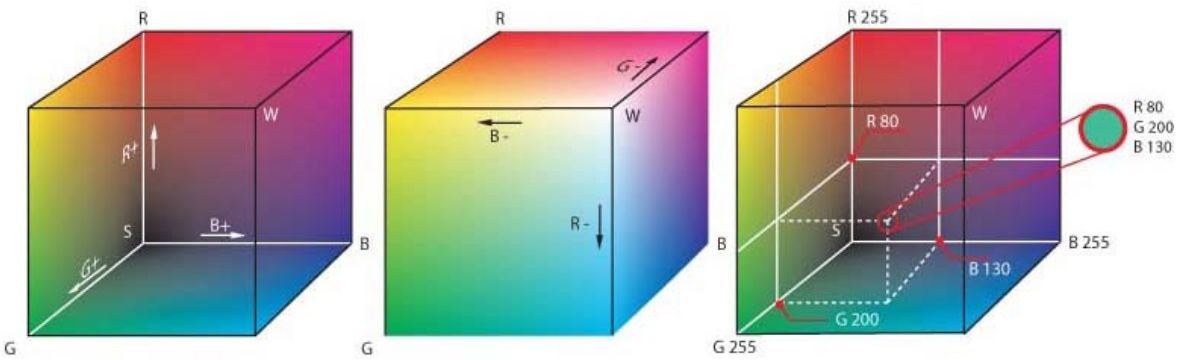
2.2.4 Barevné modely obrazu

Informace o barevnosti pixelu je obvykle reprezentována jako bod barevného prostoru. Barevný prostor je obvykle reprezentován třírozměrnou maticí, barevná vlastnost jednoho pixelu je tak reprezentována trojicí čísel. Při image processingu[4] se nejčastěji používají tyto barevné modely: [9]

[4] počítačové zpracování obrazu

2.2.4.1 RGB model

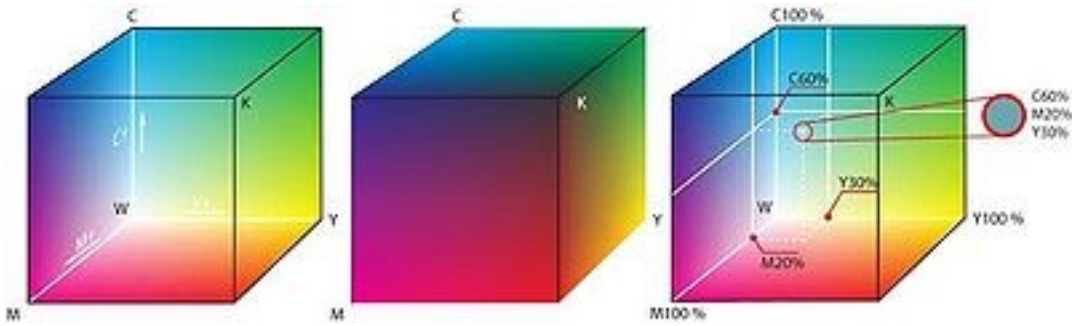
Nejrozšířenější barevný model, který se používá ve většině obrazových formátů. Jeden pixel je reprezentován trojicí hodnot odpovídající jasu červené (R), zelené (G) a modré (B). Jedná se o aditivní model[5]. Reprezentace žluté barvy v RGB modelu: R=255 G=255 B=0. [9]



Obrázek 3 - Výběr jednoho barevného bodu z třírozměrného prostoru [9]

2.2.4.2 CMYK model

Základem tohoto modelu je subtraktivní míchání barev, to znamená, že s každou další přidanou barvou se ubírá část původního světla. Výslednou barvu pak tvoří zbývající vlnové délky. CMYK model obsahuje azurovou (Cyan), purpurovou (Magenta), žlutou (Yellow) a černou (Key black). Tento barevný model se nejčastěji používá v tiskárnách, kdy výsledná barva vzniká smícháním těchto čtyř základních pigmentů. Převod mezi RGB a CMYK nelze provést úplně přesně bez jistého barevného zkreslení. [9]

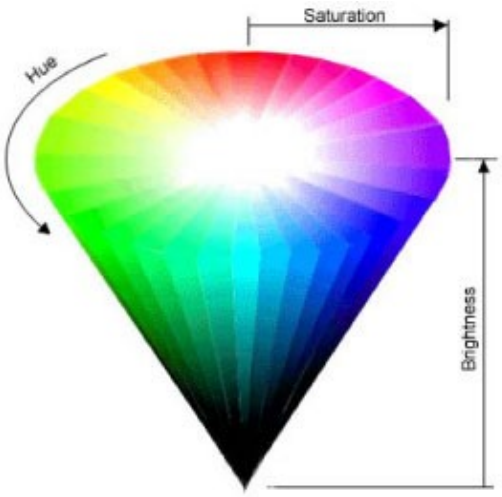


Obrázek 4 - Výběr jednoho barevného bodu [9]

2.2.4.3 HSB model

Tento barevný model nejvíce odpovídá lidskému vnímání barev, má tři základní parametry: odstín (Hue), sytost (Saturation) a jas (Brightness). Odstín se udává úhlem na kole barev (0°-360°), sytost a jas v procentech. Největší nedostatky tohoto barevného modelu plynou z této reprezentace jednotlivých barev. Základní nedostatky jsou přechod mezi černou a bílou a plynulost změny barevného tónu. Tento model je vhodný k editaci fotografií a úpravě obrazu. [9]

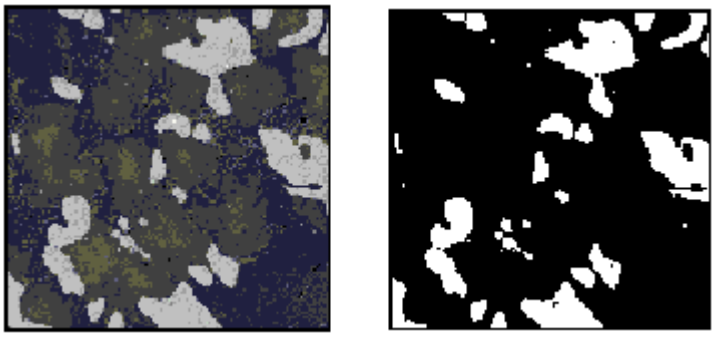
[5] vzájemné sčítání barev



Obrázek 5 - Reprezentace spektra HSB modelu [9]

2.2.5 Prahování

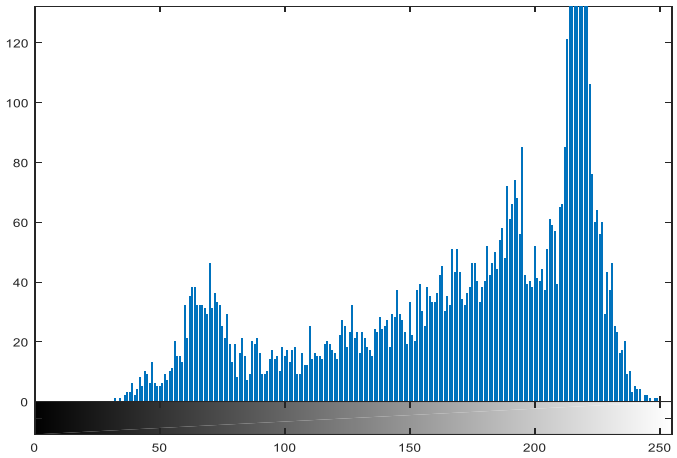
Je nejzákladnější metoda segmentace obrazu, která se zakládá na hodnocení jasu každého pixelu. Všechny hodnoty jasu nižší než stanovený práh jsou vyhodnoceny jako pozadí a vyšší hodnoty jako popředí. Nejčastěji se nepoužívá pouze jedna prahová hodnota ale více, pro lepší hodnocení nehomogenního obrazu. [8]



Obrázek 6 - Obrázek vpravo znázorňuje stav před prahováním, vlevo po prahování [8]

2.2.6 Histogram

Je graf závislosti absolutních četností v obraze v závislosti na jejich jasových hodnotách. Na ose x je vyneseno 256 sloupců odpovídající úrovním jasové škály a na ose y je znázorněn celkový počet pixelů obrazu v dané úrovni jasu. Nejčastěji používáme histogram při různých jasových transformacích a následné segmentaci obrazu. [9]



Obrázek 7 - Ukázka histogramu ze softwaru MATLAB

3 Dynamický obraz

3.1 Definice video signálu

Video signál je v podstatě posloupnost různých obrazů závislých na čase. Statický obraz je prostorové rozdělení intenzity, které zůstává v čase konstantní, zatím co obraz měnící se v čase, má prostorovou distribuci intenzity, která se mění s časem. Video signál je považován za sérii snímků tzv. obrazů. Iluze souvislého obrazu docílíme rychlou změnou jednotlivých obrazů, což je obecně nazýváno jako frame rate[6]. [2]

Při získávání digitálního videa je postup podobný jako u statického digitálního obrazu (viz kapitola 2.1). Obrazová informace je zde digitalizována jak v prostoru, tak čase a výsledný pixel intenzity je poté kvantován. Proces digitalizace můžeme vidět na diagramu, který je znázorněn na blokovém schématu (Blokové schéma 1). [12]

Blokové schéma 1 - Proces získávání video záznamu [12]

Citlivost HVS[7] se mění v závislosti na prostorové frekvenci obrazu. V praxi to znamená, že hodnota každého pixelu musí být kvantována s konečnou přesností 8 bitů na jeden jasový vzorek. Jak bylo popsáno výše, video je složeno ze sekvence obrazů, promítaných v rychlém sledu. To nám dává pocit nepřetržitého pohybu, v případě, že je odstup dvou po sobě jdoucích snímků příliš velký, divák bude vnímat pohyb jako nespojitý. Citlivost HVS klesá při vysokém počtu snímků za sekundu. V praxi běžně používaná vzorkovací frekvence je 24 fps[8] nebo vyšší. [2]

[6] Počet snímků za sekundu
[7] Human Visual Systém – lidské oko
[8] Frames per second – snímky za sekundu

Tabulka 1 znázorňuje v dnešní době nejpoužívanější video formáty, můžeme z ní vyčíst formát videa, pixelové rozlišení a nejčastější použití. [10, 11, 12]

Tabulka 1 - Základní NFTC formáty [12]

Formát	Rozlišení	Použití
Sub-QCIF	128 x 96	Mobilní multimedia
QCIF	176 x 144	Videokonference, mobilní multimedia
CIF	352 x 288	Videokonference, webkamery
4CIF	704 x 576	SDTV a DVD video
16CIF	1408 x 1152	HDTV a DVD video

3.2 Video procesing

Potřeba, jakýmkoliv způsobem zpracovávat video, rostla s nástupem novinek jako DVD, DSS[9], HDTV, digitálních fotoaparátů a digitálních kamer. Mezi základní oblasti zpracování videa patří video komprese, indexování, segmentace. [11]

3.2.1 Indexování videa

Používá se pro usnadnění vyhledávání v obrazové informaci, rozsáhlých multimediálních databázích. Pro vytvoření indexu musí být zvolena reprezentativní sada klíčových snímků, které zachycují celý videozáznam. [2]

3.2.2 Převzorkování

Základní koncept převzorkování je snížit rozměr vstupního obrazu (horizontální a vertikální) a tím i počet prvků, které mají být kódovány, ještě před kódováním samotným. Tato technika je považována za jednu ze základních kompresních technik, jelikož využívá základní fyziologii lidského oka a tím odstraňuje subjektivní redundance obsažené ve video záznamu. Lidské oko je více citlivé na změny jasu než barvy, formát RGB je zde proto nevhodný. Používá se zde formát YUV, tří vektorový formát, kde složka Y je jasová a zbylé dvě jsou složky chromatické. [10]

Převzorkování se nejčastěji označuje ve formátu *X:X:X,* první číslo je počet jasových vzorků Y, zbylé dvě čísla udávají počet vzorků chromatických složek U a V. [11]

3.2.3 Komprese videa

Kompresi videa můžeme chápat jako změnu velikosti, za účelem lepšího přenosu. Například video velikosti 176 x 144 px, 30 fps a 24 bitů na pixel by vyžadovalo přenosovou rychlost 18,25 Mbps. Efektivní komprese může být dosaženo snížením časové a prostorové redundance. Časová redundance využívá podobnosti sousedních obrazů v čase, naproti tomu redundance prostorová využívá podobnosti sousedních pixelů. Odstraněním nadbytečné informace z videosekvence snížíme i bitovou hloubku celého videostreamu. Kompresní algoritmy můžeme rozdělit do dvou skupin bezztrátové videokomprese a ztrátové videokomprese. Ztrátová komprese je nejpoužívanější u většiny multimediálních aplikací, použití bezztrátové komprese je pouze u archivace videí medicínských a satelitních snímků. [12]

[9] Digitální satelitní systém

4 Rešerše metod používaných pro detekci pohybu

Sledování objektů ve videu je proces, kdy monitorujeme určité parametry objektu (směr pohybu, orientace, okluze atd.), za účelem získání užitečných informací pro jeho sledování. Vycházíme zde z předpokladů, že objekt, který sledujeme je viditelný a jeho pohyb omezený. Můžeme sledovat široké spektrum objektů, nacházejících se na různých místech. Například sledování osob v budovách, na letištích nebo železničních stanicích. [1]

Metoda detekce pohybu z videozáznamu je v odborných článcích velice diskutované téma. V rešeršní části jsem se chtěl však zaměřit na souhrn všech dostupných metod, posouzení jejich efektivity a následný výběr a implementaci nejschůdnější metody. Tyto články jsem si vybral proto, že zahrnují různé způsoby, jak můžeme detekovat objekty ve video záznamech. Ať už je to pomocí PN typů, nebo na základě odlišné barevnosti (histogram-based), nebo pomocí odečtení pozadí (background subtraction). Všechny metody mají své výhody a nevýhody, což také zohledním v mé práci. [1]

4.1 Model prostředí

Nejdůležitější vlastností většiny metod řešících detekci pohybu je, že ke své správné činnosti potřebují model prostředí neboli pozadí scény. Na základě správného zjištění pozadí lze poté vyhodnocovat další informace. Metod pro extrakci bylo vymyšleno již velké množství, ať už jednodušších anebo komplikovanějších. Zatímco algoritmicky složitější techniky často dosahují vynikajících výsledků, jejich výpočetní náročnost je velmi kritická. Nicméně i jednoduché techniky, jako je adaptivní medián, mohou dávat velmi dobré výsledky s mnohem menší složitostí. Základní dělení technik pro modelování pozadí je dvojí: [1, 13]

4.1.1 Nerekurzivní techniky

Používá se zde technika posuvného okénka pro výpočet pozadí, pomocí něj se do zásobníku (anglicky *bufferu*) ukládá určitý počet předchozích snímků. Pozadí se následně vypočítá přes všechny obrázky v zásobníku. Tato metoda je velice adaptivní, neboť není závislá na předem daném počtu snímků a spoléhá se pouze na fixní velikost zásobníku. Na druhou stranu příliš malá fixní velikost zásobníku může způsobit zkreslení v podobě špatného výpočtu modelu pozadí. Mezi nerekurzivní techniky řadíme například metodu mediánu a metodu průměru. [1, 13]

4.1.1.1 Metoda průměru

Jelikož se jedná o jednu z nejjednodušších metod tvorby modelu prostředí, její výpočet je velice jednoduchý. Všechny pixely modelu se rovnají průměrné hodnotě všech pixelů v zásobníku. Metoda není příliš účinná s velkou citlivostí na prahování. [1, 13]

4.1.1.2 Metoda mediánu

Základ této metody spočívá ve zprůměrování každého pixelu v obraze pomocí mediánu přes celý zásobník. Postup je aplikován na každý barevný kanál zvlášť. Souřadnice pixelu si označíme (i, j) a extrahovaný model pozadí $BG_{ij} = \{R_{BG}, G_{BG}, B_{BG}\}$ je získán:

$$B_{i,j} = \begin{cases} R_{bg} = Median\,\{R_1, R_2, \dots R_t\} \\ G_{bg} = Median\,\{G_1, G_2, \dots G_t\} \\ B_{bg} = Median\{B_1, B_2, \dots B_t\} \end{cases} \quad (2)$$

Metoda funguje velice spolehlivě má však zvýšenou jasovou citlivost a při změně jasových podmínek scény je nutný přepočet celého modelu pozadí. [1]

4.1.2 Rekurzivní techniky

Tato metoda nepotřebuje ke své činnosti zásobník, jelikož pracuje pouze s jedním aktuálním snímkem a ten je aktualizován při každém přechodu nového snímku. Snímky z dávné minulosti tak mohou mít vliv na aktuální stav modelu scény. Paměťové nároky, oproti metodám nerekurzivním, jsou menší, avšak tato výhoda má za následek promítnutí chyb z dávné minulosti do aktuálního snímku. Mezi nerekurzivní techniky patří například metoda směsi Gaussiánů. [1]

4.2 ROI[10] metoda

Navrhovaný algoritmus je dán následujícími kroky:

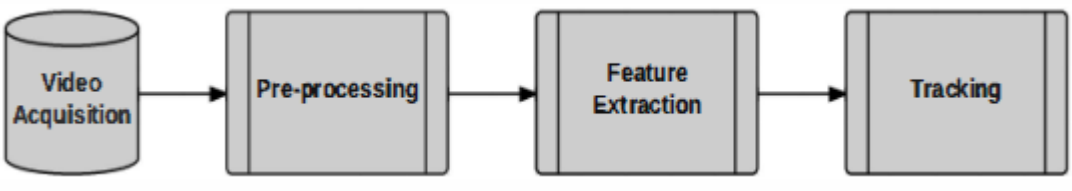
1. Nasnímáme video z kamery do systému
2. Ořežeme kopii oblasti nebo objektu, který budeme sledovat
3. Vytvoříme ROI v unifikované vzdálenosti 20px od objektu
4. Vytvoříme pole s velikostí 50 obrazů
5. Aplikujeme Template matching algorithm, abychom získali polohu objektu
6. Uložíme extrahovaný obraz jako první položku pole
7. ROI je v poli uložen s indexem 0
8. Porovnáme extrahovaný obraz s ROI obrazem
9. Jestliže je nalezena shoda, aplikujeme metodu PCA, abychom detekovali cílový objekt. Sledujeme procentuální shodnost.

Při rychlém snímání videa může dojít ke změně vzhledu objektu a může se stát, že objekt už dále nebudeme schopni správně detekovat. Autoři článku proto popisují PN Learning Algorithm. Funguje na principu, že objekt, který sledujeme je P-typ a pozadí za objektem je rozděleno na stejný počet N-typů. P-typy se stanou referenční a jsou porovnávány se vstupním obrazem. Porovnává se zase procentuální podobnost buď s N-typem, nebo P-typem. Pokud je vstupní obraz shodný s N-typem, je okamžitě vyřazen z detekce. Všechny vstupní obrazy shodné s P-typem jsou opět uloženy do pole. [20]

[10] Region of interest

4.3 Algoritmus detekce pohyblivých objektů a jejich tras z videa pomocí jejich barevných vlastností

V prvním kroku je zvolena pozice šablony T v prvním snímku videosekvence. Zvolíme h_T jako histogram šablony T a počet pixelů T jako $|T|$, což je rovno součtu nad jednotlivým polem histogramu. Za druhé budeme pokládat frame[11] jako dotaz a histogram-based algorithm[12] najde umístění objektu, přes vyhodnocení míry podobnosti histogramu šablony a histogramu z videosekvence.



Blokové schéma 2 - Blokové schéma algoritmu detekce pohybu pomocí barevných vlastností

Obrázek 8 - Výsledky sledování pohybu hrnečku

V prvním kroku získáme videozáznam. Poté v pre-procesingu se záznam nasnímkuje na jednotlivé obrazy, ze kterých na základě odlišeného barevného spektra probíhá kapková analýza. V posledním kroku tzv. tracking se daný objekt na základě rozdílných vlastností detekuje. V případě pohybu analyzovaného předmětu se simultánně mění poloha detekované oblasti. Touto procedurou jsme schopni provést detekci objektů v reálném čase. [21]

4.4 Algoritmus detekce pohybujících se objektů z video záznamu

- **Background subtraction method**[13] - metoda je používána pouze u statických kamer a funguje na základě rozdílu mezi aktuálním a referenčním snímkem, který nazýváme „snímek/obraz pozadí“
- **Frame differencing method**[14] **-** tato metoda využívá malého rozdílu dvou nebo tří po sobě jdoucích snímků v obrazové sekvenci, za účelem detekce pohybu ve videu
- **Optical flow method**[15] **-** přístup metody je takový, že prvky každého snímku jsou sledovány za určitý časový úsek, aby bylo možné stanovit rychlost a vektor rychlosti pohybujících se objektů ve scéně.

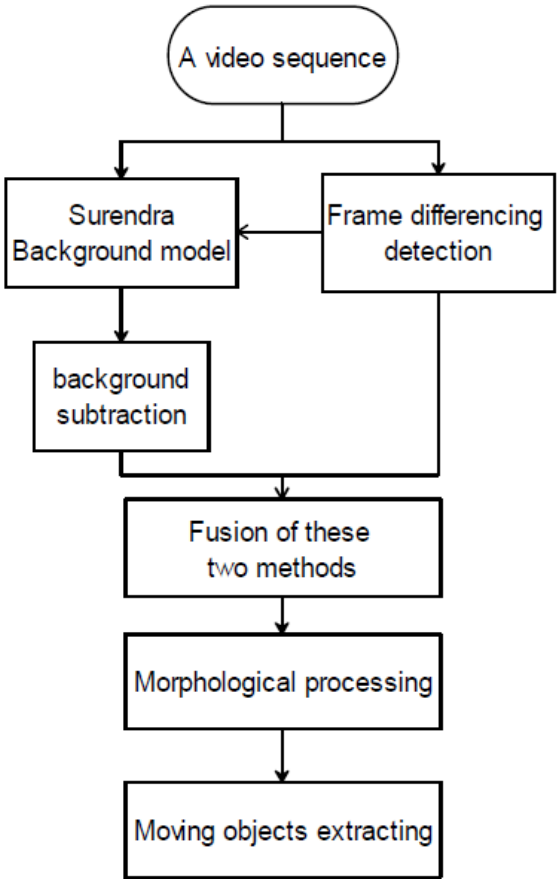
[11] Snímek
[12] Algoritmus na bázi histogramu
[13] Metoda odebrání pozadí
[14] Metoda rozdílných snímků
[15] Metoda pohybu vektorového pole

Článek na úvod srovnává již existující metody (Tabulka 2), shrnuje všechny výhody a nevýhody.

Tabulka 2 - Porovnání tří základních metod

Metoda	Výhody	Nevýhody
Background subtraction	Rychlé, snadné, přesné	Náchylné ke změnám osvětlení a pozadí
Frame differencing	Vysoce přizpůsobivé k rychlým změnám obrazu	Nevhodné pro statický obraz
Optical flow method	Přesné	Složité na implementaci

Poté autoři článku přichází s vlastním návrhem algoritmu, který se skládá z hlavních pěti kroků, které jsou Surendra background model[16], background subtraction, frame differencing, extrakce pohybujících se objektů z obrazu, získaných kombinací těchto dvou algoritmů. Používá se Surendra background algoritmus a frame differencing pro jeho aktualizaci, poté se extrahuje pohybující objekt z fúze obrazů získaných z těchto dvou metod.



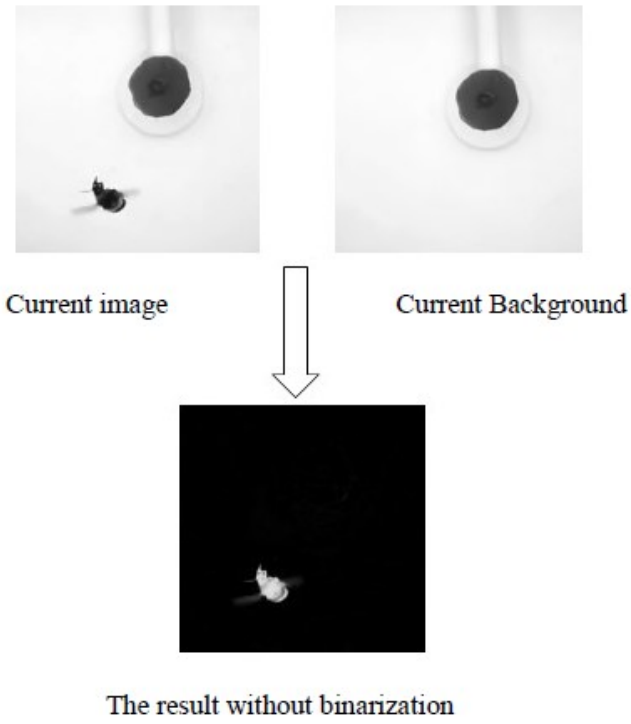
Blokové schéma 3 - Funkční schéma navrhované metody

[16] Surendrova metoda odebrání pozadí

V této metodě je používána Surendra background model. U této metody, předpokládáme, že všechny snímky, které zpracováváme, jsou snímány kamerou. Pak I_k a B_k jsou intenzity daného snímku a jeho pozadí, na základě pořadí. Rozdílový obraz $D_k(x, y)$ používaný k detekci pohyblivých objektů může být definován následovně:

$$D_k(x,y) = \begin{cases} 1 & |I_k - I_{k-1}| > T_d \\ 0 & |I_k - I_{k-1}| \geq T_d \end{cases} \quad (3)$$

Kde T_d je předem definovaná prahová hodnota a volí se jako rozdíl obrazu $D_k(x, y)$. Hodnota $D_k(x, y)$ rozhoduje, zda se obraz pozadí změnil či nikoliv.



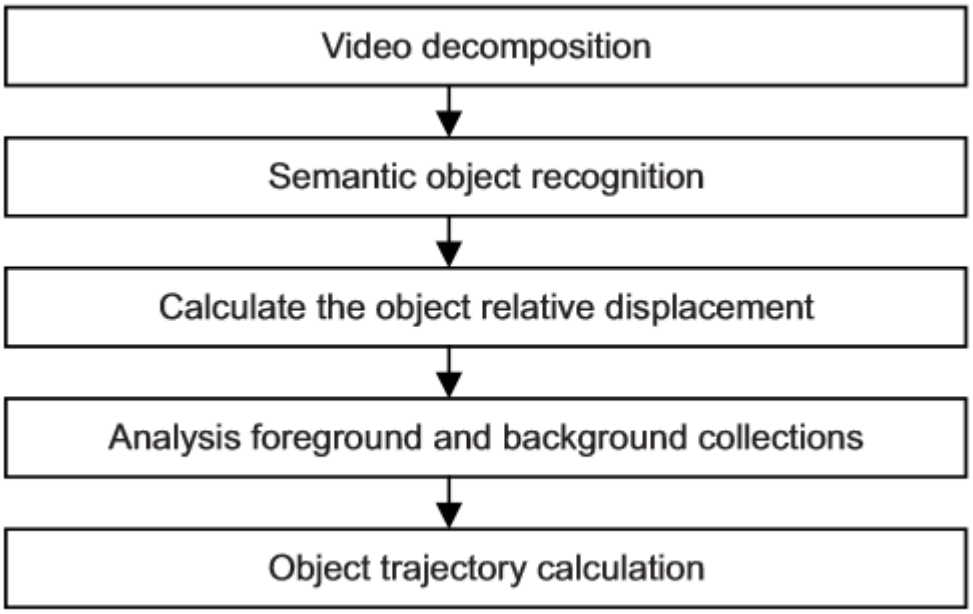
Obrázek 9 - Výsledky metody bez binarizace

Jak můžeme vidět na obrázku (Obrázek 9), po použití Surendrovi metody a metody extrakce objektu, potřebujeme snímek binarizovat. I přesto se může stát, že nám ve snímku zůstane spousta nežádoucích jevů, jako je šum nebo skvrny. Takže použijeme morfologické filtry pro jejich odstranění. Mezi nejčastější techniky patří dilatace a eroze obrazu. Dále můžeme použít další operace jako segmentace obrazu, extrakce příznaků, detekce hran, filtrace obrazu.

Autoři článku metodu implementovali na videozáznam s letící včelou. Detekce byla úspěšná, což jen dokazuje kvalitu popisované metody, kdy jsme schopni sledovat malé objekty i za velkých rychlostí. Tento nově navržený algoritmus používá Surendrovu metodu odebrání pozadí a metodu rozdílných snímků pro porovnání dvou obrazů a následnou aktualizaci. S obrazů získaných z těchto dvou metod se následně detekuje pohybující se objekt. [22]

4.5 Algoritmus detekce pohybujících se sémantických objektů z video záznamu

Navržený algoritmus je založený na relativním umístění objektu. V souladu se zásadou o relativním pohybu objektů můžeme vypočítat vzdálenost mezi nimi. V závislosti na čase se vzdálenost mezi jednotlivými objekty mění. Mezi některými zůstává konstantní a mezi některými se zvětšuje. Díky tomu jsme schopni vytvořit skupinu objektů, jejichž relativní posun je malý (pozadí). A druhou skupinu objektů s opačnými vlastnostmi (popředí). Sledováním vzdálenosti mezi těmito dvěma skupinami určíme trajektorii pohybu.



Blokové schéma 4 - Proces detekce trajektorie

I. *Rozpoznávání objektů*

Extrakci objektů můžeme shrnout do následujících třech kroků:

- **Segmentace videa:** Videozáznam je rozdělen do několika fragmentů, informace z těchto rozdělených fragmentů jsou uloženy do databáze.
- **Rozpoznávání objektů:** Na základě rozdílné textury a obrysů je záznam rozdělen do nepravidelně tvarovaných regionů; každý region je viděn jako sémantický objekt
- **Přiřazení sémantiky**

II. *Detekce a trakování sémantických objektů*

Následující tabulky zobrazují informace o objektu a jejich možné uložení do databáze.

Tabulka 3 - Návrh tabulky pro uložení segmentů do databáze

Proměnná	Datový typ	Konstanta	Popis
segmentid	int	key	Id segmentu
description	varchar		Popis segmentu
startframe	int		Id prvního snímku
endframe	int		Id posledního snímku

Tabulka 4 - Návrh tabulky pro uložení objektů do databáze

Proměnná	Datový typ	Konstanta	Popis
objectid	int	key	Id objektu
fatherid	int		Id vzdáleného objektu
feature	binnary		Rys objektu
segmentid	int	foreing key	Id segmentu, ve kterém se objekt nachází
description	varchar		Popis objektu

Tabulka 5 - Objekt – pozice

Proměnná	Datový typ	Konstanta	Popis
frame	int	key	Id snímku
objectid	int		Id objektu
Xposition	int		X-ová souřadnice středu objektu
Yposition	int		Y-ová souřadnice středu objektu

Další tabulka (Tabulka 6) je velice důležitá pro pozdější analýzu trajektorie objektů. Obsahuje všechny objekty v záznamu a pozice jejich středů. Tato tabulka slouží k analýze pohybu objektů.

Tabulka 6 - Posun objektu vektorově

Proměnná	Datový typ	Konstanta	Popis
iFrameid	int	key	Id snímku i
jFrameid	int		Id snímku j
iobjectid	int		Id objektu i
jobjectid	int		Id objektu j
Xdistance	int		X-ová souřadnice vektoru posunu
Ydistance	int		Y-ová souřadnice vektoru posunu

Odborný článek dále řeší praktické využití a je zde popsán experiment s video sekvencí letícího letadla. Algoritmus detekuje pět objektů a 4 snímky z celé videosekvence, na kterých je tento experiment řešen. Na závěr článku je zobrazena tabulka (Tabulka 7) kde vidíme pohyb letadla (jobjectid = 4), vzhledem k segmentu mraku (iobjectid = 1). [23]

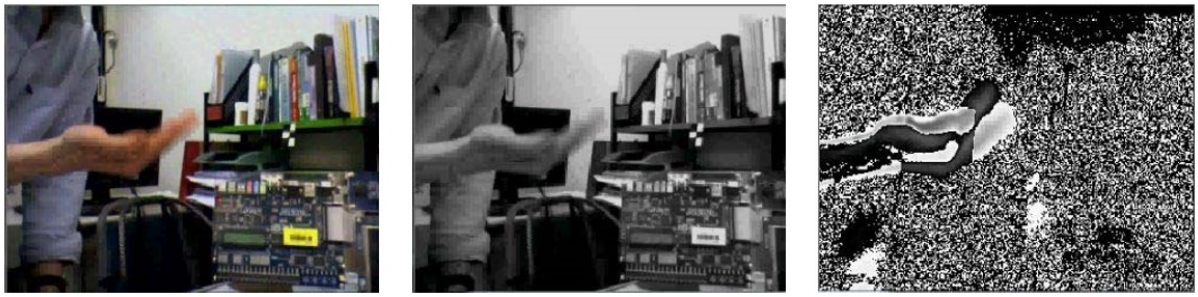
Tabulka 7 - Výsledná tabulka pohybu letadla

frameid	iobjectid	jobjectid	xdistance	ydistance
1	1	4	−250	115
2	1	4	−66	116
3	1	4	39	96
4	1	4	116	85

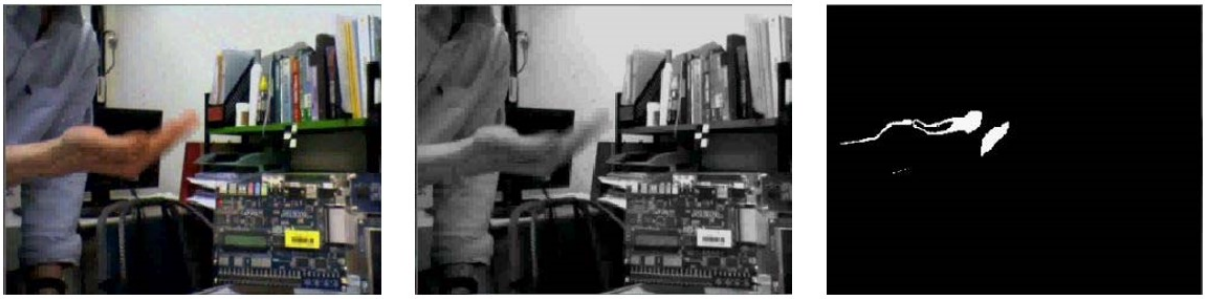
4.6 Rozpoznávání směru pohybu objektů na základě historických trajektorií

Tento odborný článek jsem si vybral, protože dobře shrnuje a popisuje část preprocesing, časovou diferenciaci a metodu mediánové filtrace.

Nejprve převedeme barevné spektrum RGB do YCbCr prostoru. V dalším kroku od sebe odečteme dva po sobě jdoucí snímky pomocí časové diferenciace (diferenciace probíhá snímek po snímku) k určení možných oblastí pohybujících se objektů. Jak můžeme vidět na obrázku (Obrázek 10). První snímek je původní obraz, druhý je jeho převedení do prostoru Y a třetí je po časové diferenciaci.



Obrázek 10 - Poslední snímek je zašuměný, bez použití mediánové filtrace



Obrázek 11 - Poslední snímek, po použití mediánové filtrace

Použití mediánové filtrace můžeme vidět na obrázku (Obrázek 11). Oproti obrázku (Obrázek 10) nevidíme zašumění způsobené odlesky prostředí. Pro účely experimentu autoři definují vzorec pro mediánovou filtraci: [19]

$$frame_{diff(x,y)} = \begin{cases} 255, if\ 200 \geq frame_{diff(x,y)} \leq 120 \\ 0, otherwise \end{cases} \quad (4)$$

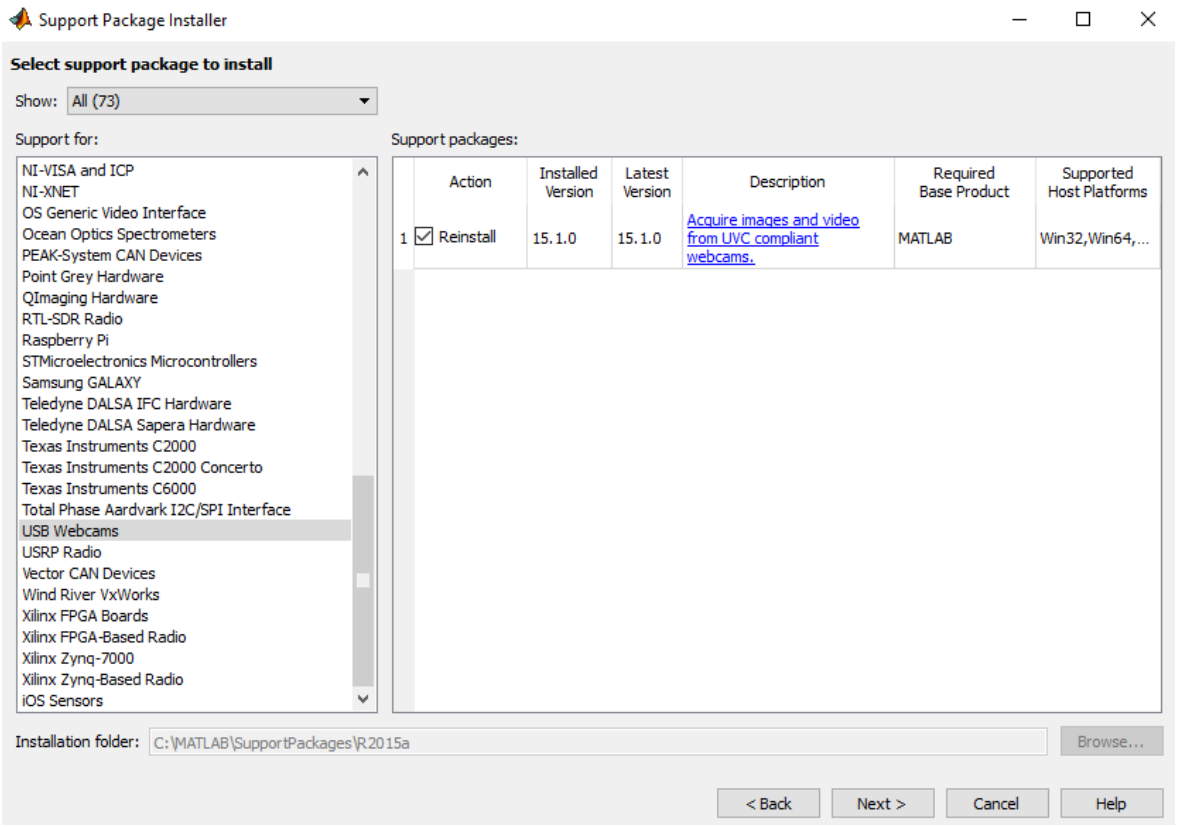
$frame_{diff(x,y)}$… hodnota pixelu po časové diferenciaci

5 Funkce softwaru MATLAB pro zpracování videa

Tato kapitola byla věnována základům práce se softwarem MATLAB, jeho propojení s externí (webovou) kamerou a základním funkcím pro práci se záznamy z připojených kamer. Tato kapitola zároveň slouží jako učební text pro osvojení si základů práce s videem v softwaru MATLAB. [14, 15, 16]

5.1 Použití MATLAB Support Package for USB Webcams a jeho funkcí

Jedním ze způsobů jak připojit webovou kameru je instalace Support Package přímo v MATLABU v záložce **Add-Ons**, **Get Hardware Support Packages.** Jak je uvedeno na obrázku (Obrázek 12). Po automatické instalaci balíčku nám byly zpřístupněny všechny funkce, které tento package nabízí. [17, 18]



Obrázek 12 - Instalace MATLAB Support Package for USB Webcams

5.1.1 Webcamlist

Funkce *webcamlist()* slouží k ověření správného propojení MATLABU a externí hardwarové kamery. Funkce vrací hodnotu všech dostupných externích kamer, které jsou připojeny. Níže můžeme vidět konkrétní příklad, zde je návratovou hodnotou funkce řetězec, datového typu char, který obsahuje název integrované kamery *Lenovo EasyCamera*.

```
ans =

'Lenovo EasyCamera'
```

5.1.2 Webcam

Funkce *webcam()* vyžaduje jeden parametr v závorce, a to číslo které udává pořadí kamery, z funkce *webcamlist()*, kterou chceme připojit. Funkce zároveň spustí danou kameru a při uložení do proměnné s touto kamerou můžeme dále pracovat. Příkaz při jeho spuštění v cmd window, vypíše základní parametry kamery, které můžeme později změnit.

```
webcam(1)

ans =

  webcam with properties:

                      Name: 'Lenovo EasyCamera'
                Resolution: '1280x720'
    AvailableResolutions: {'1280x720'  '640x480'  '320x240'
'160x120'  '800x600'  '640x360'  '1280x720'}
                       Hue: 50
                  Contrast: 50
                Saturation: 50
                Brightness: 50
                 Sharpness: 50
                     Gamma: 50
```

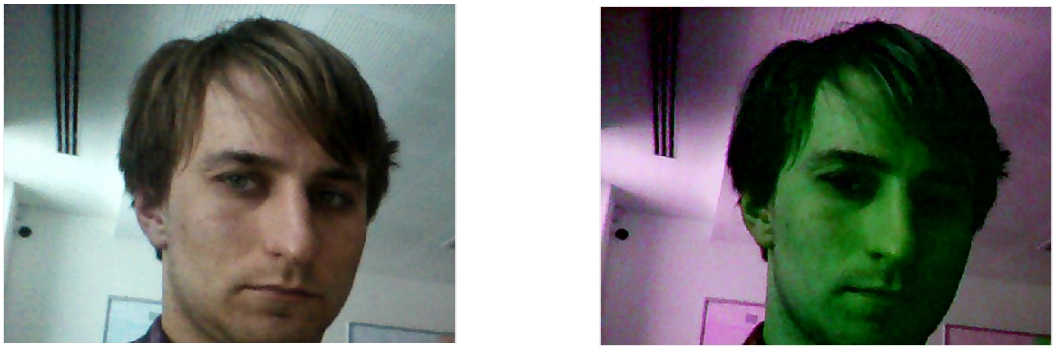
Příkaz *webcam(1)*, jež je uveden výše vypíše do comand window tyto parametry:

- **Name –** Název kamery, tento název je systémový a nelze změnit.
- **Resolution –** Aktuální rozlišení které je nastaveno, parametr lze změnit pomocí příkazu *cam.Resolution = '640x480'*.
- **AvailableResolutions** – Dostupné rozlišení, na které můžeme změnit parametr *resolution*, odpovídá hardwarové specifikaci dané kamery
- **Hue –** Pomocí tohoto parametru může být nastaven odstín obrazu snímaného kamerou. Rozsah může být nastaven od 0 do 100. Defaultní hodnota je 50.
- **Contrast –** Nastavuje kontrast v obraze. Rozsah od 0 do 100. Defaultní hodnota je 50.
- **Saturation –** Nastavuje množství použité barvy v obraze, saturaci. Rozsah parametru může být od 0 do 100. Defaultní hodnota je 50.
- **Brightness –** Změna hodnoty jasu použitá v obraze. Rozsah od 0 do 100. Defaultní hodnota je 50.
- **Sharpness –** Umožnuje systémové nastavení ostrosti obrazu. Rozsah od 0 do 100. Defaultní hodnota je 50.
- **Gamma –** Umožňuje nastavit gama korekci v rozsahu od 0 do 100. Defaultní hodnota je 50.

Příklad změny parametrů kamery před samotným snímáním pomocí funkce *webcam()*. Záznam z kamery je zobrazován pomocí funkce *preview()* v grafickém okně. V případě, že chceme, aby prohlížení bylo zavřeno, použijeme příkaz *clear cam.* Na obrázku (Obrázek 13), můžeme vidět nativní snímek (vpravo) a snímek po změně parametrů (vlevo).

```
cam = webcam(1)

cam.Resolution = '640x480'
```

```
cam.Saturation = 80
cam.Hue = 20
cam.Brightness = 30
cam.Contrast = 60
cam.Sharpness = 90
cam.Gamma = 50

preview(cam)

clear cam
```



Obrázek 13 - Náhled na nativní snímání webkamerou pomocí funkce *preview()* (vpravo). Náhled na snímání webkamerou po změně parametrů, viz kód (vlevo).

5.2 Nahrávání videa

Následující kód byl vytvořen jako ukázka jednoduchého programu pro zachycení video záznamu v MATLABU. Nevýhodou bylo, že při použití musel být nastaven přesný počet snímků, které jsme chtěli zaznamenat. Také nebylo možno přesně nastavit vzorkovací frekvenci a periodu, se kterou bude daný video soubor uložen.

Příkaz *VideoWriter()* vytvoří avi soubor, do kterého bude dále zapisováno. Celý tento soubor je uložen do proměnné *vidWriter* a otevřen pomocí funkce *open(vidWriter)*. Následný cyklus při každé jeho iteraci provede zachycení snímku funkcí *snapshot()* a uložení zachyceného snímku do souboru funkcí *writeVideo(vidWriter, img)*. Funkce má dva parametry, první je soubor, do kterého je prováděn zápis, druhý parametr je pak zapisovaný snímek. Po provedení všech iterací je soubor zavřen funkcí *close()* a webkamera vypnuta funkcí *clear vid*.

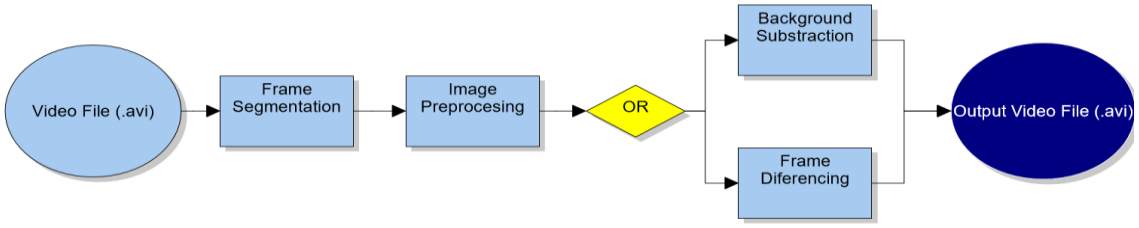
```
webcamlist;
vid = webcam(1);
vidWriter = VideoWriter('frames.avi');
open(vidWriter);

for iFrame = 1:100

    preview(vid);

    img = snapshot(vid);

    writeVideo(vidWriter, img);

end
close(vidWriter);
clear vid
```

6 Praktická část

V praktické části mé bakalářské práce byla věnována pozornost teoretickému návrhu algoritmu, použitým metodám, praktickému naprogramování algoritmu, testování algoritmu, zhodnocení výsledku přesnosti detekce. Data, která jsou využívána během vývoje a testování algoritmu, byla experimentálně vytvořena pouze za účelem testování navrženého algoritmu. Snímaná místnost byla chodba v bytě, po které se pohybují osoby, jejichž pohyb byl detekován.

6.1 Návrh algoritmu pro detekci pohybu

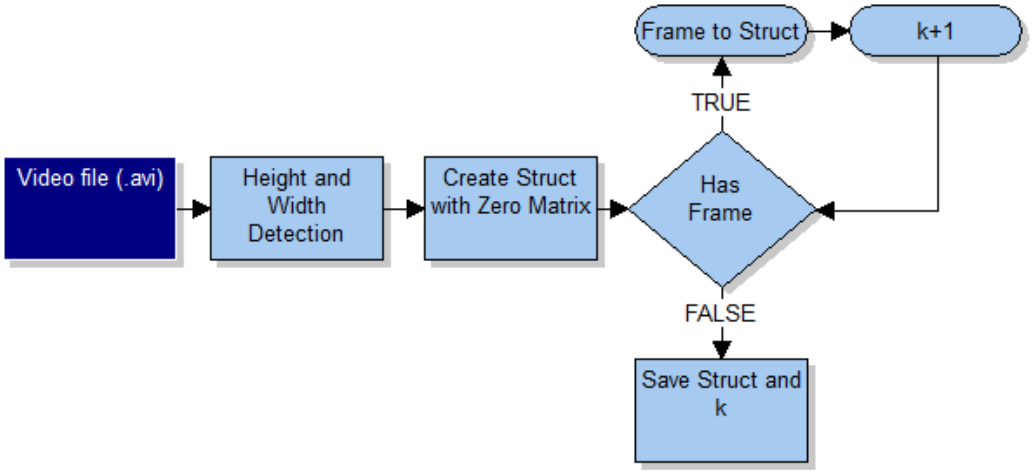
Při návrhu algoritmu bylo vycházeno z předpokladu, že testovaná data budou pořízena statickou kamerou, bylo tedy snímáno stále jedno místo. Testovaný algoritmus fungoval na principu analýzy již existujících záznamů, na vstupu byl tedy video soubor ve formátu (.avi). Vstupní video záznam musel být nejprve rozsegmentován na jednotlivé snímky (Frame segmentation). Po správné segmentaci a uložení jednotlivých snímku došlo k důležité části předzpracování jednotlivých snímků (Image preprocesing). Následně bylo možné zvolit metodu pro detekci pohybu (Background substraction, Frame diferencing). Poslední částí algoritmu bylo uložení výstupu do video soboru ve formátu (.avi). Návrh algoritmu můžeme vidět na blokovém schématu (Blokové schéma 5).



Blokové schéma 5 - Schéma navrhovaného algoritmu

6.1.1 Segmentace

Vstupní video bylo chápáno jako 4-D matice, viz kapitola 3.1. Při genezi video záznamu mě zajímaly tyto stěžejní parametry: výška, šířka a počet snímků. Před samotnou segmentací video sekvence nebyl znám přesný počet snímků, které video může obsahovat, bylo vhodné zvolit pro uložení jednotlivých snímků datový typ struktura. Výhodou tohoto datového typu je, že nemá předem definovanou délku. Jeden prvek struktury tedy bude obsahovat, výšku a šířku obrazu, a jeho tři chromatizační složky (modrou, zelenou a červenou). Je zde použita funkce *VideoReader()*, která otevře vybraný video soubor pro čtení a uloží ho do proměnné. Tato funkce má naimplementované příkazy pro lepší manipulaci ze souborem např. *hasFrame()* a *readFrame()*. Tyto funkce nám usnadňují práci při čtení z video souboru.



Blokové schéma 6 - Schéma pro načtení a segmentaci obrazu z video záznamu

Výše uvedené blokové schéma (Blokové schéma 6) popisuje proces segmentace jednotlivých snímků z video záznamu. Soubor byl nejprve načten funkcí *VideoReader()*, poté byla detekována výška a šířka obrazu pomocí funkcí *vidObj.Height* a *vidObj.Width*. V třetím kroku dílčího algoritmu byla vytvořena struktura pro uložení jednotlivých snímků. Nejdůležitějším prvkem celého algoritmu bylo vytvoření cyklu pro načtení snímků z videa. Funkcí *hasFrame()* byly načítány jednotlivé snímky z video souboru, ty byly poté pomocí funkce *readFrame()* ukládány do struktury. Po skončení celého cyklu načítání snímků byla na výstupu struktura vlastních framů a proměnná obsahující hodnotu rovnající se počtu snímků.

6.1.2 Předzpracování obrazu

V této fázi algoritmu nám již byly známy všechny parametry obrazu, potřebné pro jeho další zpracování, to je jaké má daný snímek rozlišení a jaká je frekvence jeho snímkování (frame rate). Tyto parametry, byly zjištěny v předchozí kapitole 6.1.1, jsou uloženy v proměnné *fram_num* (frame rate) a struktuře *frame* (jednotlivé snímky). Stále byl však obraz pro následné zpracování příliš velký a jeho uložení ve struktuře není příliš vhodné. Obě používané metody pracují totiž s maticovým zápisem jednotlivých obrazů. Ve fáze předzpracování byl obraz převeden z RGB spektra (více v kapitole2.2.4.1) do spektra odstínů šedi. Ze čtyřrozměrné matice, jejíž jednotlivé vrstvy jsou R, G, B a počet snímků, tak byla vytvořena matice dvojrozměrná, která obsahuje pouze obrazovou stopu a počet snímků. Touto procedurou byl obraz převeden do stavu, který byl vhodnější pro pozdější detekci pohybu.

```
%%Convert RGB to Grayscale%%
for iFrame = 1:frameNum
    % Read image from struct and write to img
    img = struct(iFrame).frame;
    % RGB to grayscale and write frame to matrix
    frame_mat(:,:,iFrame) = rgb2gray(img);
end
```

Obrázek 14 - Ukázka kódu pro převod RGB obrazu do šedobarevného obrazu

6.1.2.1 Geneze monochromatického záznamu

Funkce *rgb2gray* převádí truecolor image[17] na šedo barevný obraz tak, že potlačí jeho složku odstínu a sytosti za účelem zachování jasu. Nejjednodušší a nejméně přesná metoda je metoda průměru. Za předpokladu že každý pixel je reprezentován trojicí hodnot (R, G, B). Šedo barevnou podobu konkrétního pixelu m, n získáme, jako podíl součtu hodnot všech složek k počtu složek viz rovnice 5.

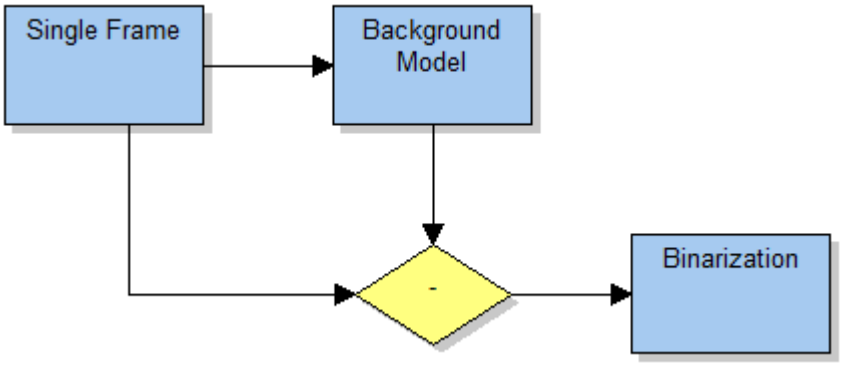
$$GrayPixel_{(m,n)} = \frac{p_{(m,n)}(R) + p_{(m,n)}(G) + p_{(m,n)}(B)}{3} \tag{5}$$

Luminiscenční metoda je oproti tomu daleko sofistikovanější, vychází z předpokladu, že lidské oko je daleko citlivější na zelenou složku spektra. Proto upravený vzorec vypadá takto viz rovnice 6.

$$GrayPixel_{(m,n)} = \frac{0{,}21 \cdot p_{(m,n)}(R) + 0{,}72 \cdot p_{(m,n)}(G) + 0{,}07 \cdot p_{(m,n)}(B)}{3} \tag{6}$$

6.1.3 Background substraction

Jádrem celého algoritmu se stala samotné detekce. V rešeršní části (kapitola 4) jsem se zabýval nejpoužívanějšími metodami detekce. Jako jedna ze dvou nejvhodnějších metod, byla zvolena metoda odečtení pozadí. Tato metoda je vhodná pro použití na záznamy ze statických kamer a není příliš výpočetně náročná. Výpočetní náročnost závisí na velikosti vstupního záznamu.



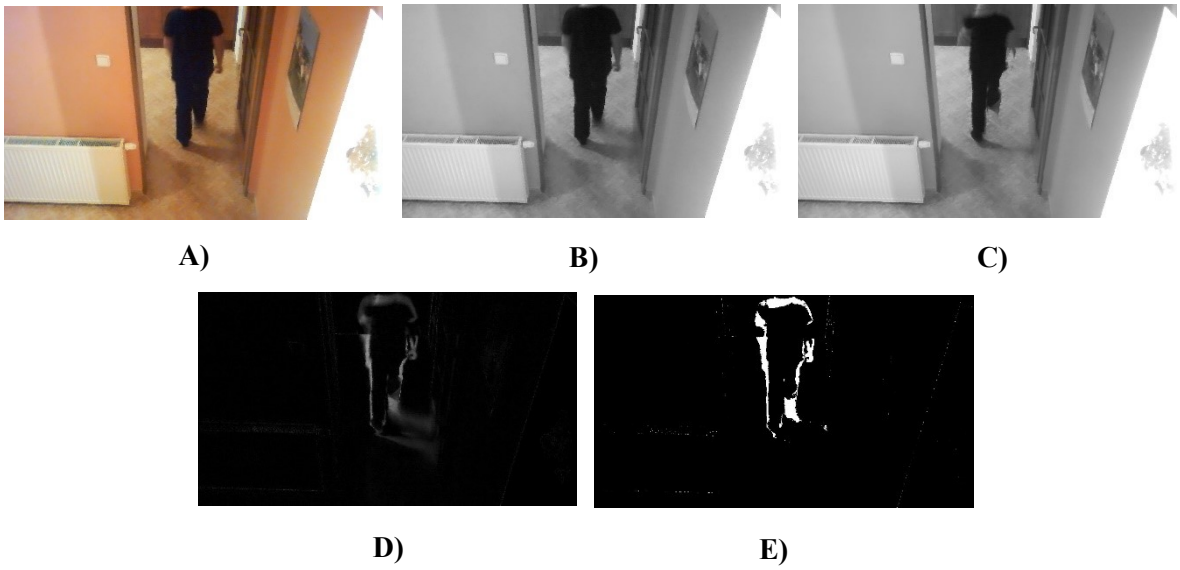
Blokové schéma 7 - Blokové schéma metody background substraction

Jak může být patrné z blokového schématu (Blokové schéma 7), inicializační fází metody je geneze modelu prostředí, který je následně odečten od každého snímku jednotlivě. Model pozadí byl vytvořen z monochromatického obrazu (Obrázek 15 b)). Cílem bylo najít maximální hodnotu pro každý pixel, mezi sousedními snímky v každém obraze. Proto byla použita funkce *imdilate()*, která provede morfologickou dilataci obrazu. Pokud použijme správně orientovaný objekt, v našem případě jednotkovou matici s jedním řádkem jedním sloupcem a rozsahem přes pět framů. Další rozměr jednotkové matice určuje, z kolika po sobě jdoucích snímků se model bude tvořit, číslo pět bylo zvoleno jako optimální po sérii testování. Čím by počet snímků byl větší, tím by byl i model pozadí přesnější. Při detekci pozadí z většího počtu snímků bychom se však dopouštěli, veliké nepřesnosti v detekci. Model pozadí pro aktuální snímek je zobrazen na obrázku (Obrázek 15 c)).

[17] Truecolor image – 24bitová barva, každá složka (RGB) má přesně 8bitů

Další části kódu se již zabývají samotnou detekcí. Jestliže máme snímek pozadí a snímek originální, při jejich odečtu dostaneme snímek obsahující pouze objekty, které byly v pohybu. Použita byla funkce i*mabsdiff()*, která provede absolutní rozdíl. Rozdíl obou snímků je vidět na obrázku (Obrázek 15 d)).

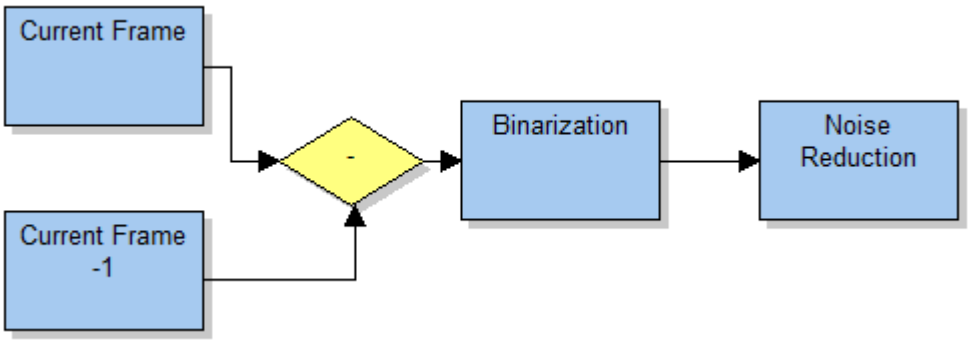
Posledním krokem celé detekce byl převod na binární obraz. Každému pixelu byla přiřazena buďto logická 1, nebo logická 0. Výsledkem byl binarizovaný obraz, kde černá barva reprezentuje statickou část a bílá barva objekty v pohybu (Obrázek 15 e)).



Obrázek 15 - a) nativní obraz b) monochromatický obraz c) snímek pozadí d) obraz po odečtení aktuálního snímku od snímku pozadí e) binární obraz

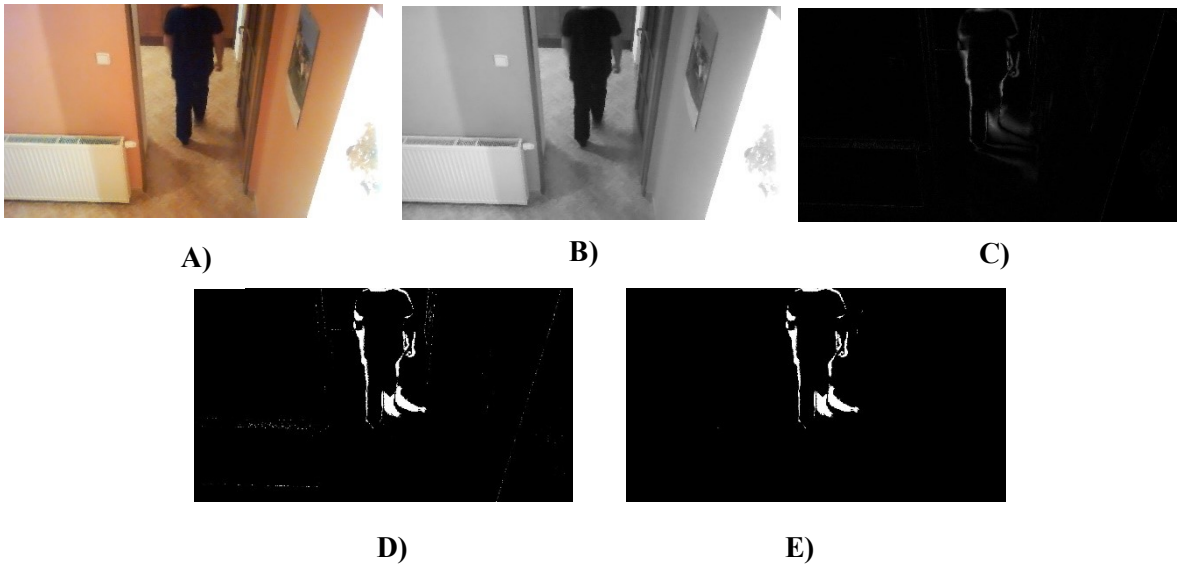
6.1.4 Frame differencing

Další metoda, která byla zkoumána v praktické části mojí práce, byla metoda snímkové diference. Jedná se o metodu, ve které byly od sebe odečteny dva po sobě následující snímky. Při odečtení dvou snímků dostaneme jejich rozdíl. Byly-li splněny předpoklady, že snímky na sebe bezprostředně navazují, statické části obrazu tak byly odečteny a výstupem byl pak obraz detekce pohybu. Tuto metodu bylo možné použít pouze u statických kamer, respektive jejich záznamu. Při pohybu kamerou by mohlo dojít k změně scény a to by mělo za následek neúplné odečtení obrazů. Jak je patrné z blokového schématu (Blokové schéma 8), obě metody si byly hodně podobné. Byla zde použita i stejná funkce pro absolutní odečet obou obrazů *imabsdiff()*.



Blokové schéma 8 - Blokové schéma metody frame differencing

Cyklus odečítání postupuje od konce, tímto bylo zabráněno tomu, aby se první snímek odečítal od nultého, který neexistuje. Zároveň však tento přístup způsobil, že výsledná matice snímků, čítala o jeden snímek méně. Obrazová informace zde tedy byla ztracena. V kontextu jednotlivých záznamů, které obsahují okolo 500 snímků, byla tato ztráta zanedbatelná. Snímek po odečtu je vidět na obrázku (Obrázek 16 c)).



Obrázek 16 - a) nativní obraz b) monochromatický obraz c) snímek pozadí d) obraz po odečtení aktuálního snímku od snímku pozadí e) binární obraz

Stejně jako u předchozí metody bylo nutné výsledný obraz binarizovat. Jak můžeme vidět na obrázku (Obrázek 16 d)), snímek po binarizaci obsahuje zašuměná místa, které se zobrazují jako bílé tečky v obraze. Toto zašumění je artefaktem detekce a je pro nás nežádoucí. Byl proto aplikován filtr, pomocí funkce *bwareaopen()*, který potlačí v binárním obraze všechny objekty větší, než je určitá prahová hodnota. V tomto případě byla použita hodnota 20px. Další parametr funkce udává nejmenší počet sousedních pixelů, které sousedí s pixelem, jenž bude potlačen. Filtrovaný obraz je demonstrován na obrázku (Obrázek 16 e)).

6.1.5 Uložení obrazu do video souboru

Posledním krokem celého algoritmu bylo uložení výsledného obrazu do video souboru ve formátu avi. To bylo realizováno pomocí cyklu, který prochází jednotlivé snímky a postupně je ukládá do video souboru. Ukázka takového cyklu je na obrázku (Obrázek 17).

```
%%Save frames to video file%%
%create video file
vidWriter = VideoWriter('vid_after_detection.avi');
%open video file for writing
open(vidWriter);
for iFrame = 1:frameNum
    %Recode to UNIT8
    frame = im2uint8(frame_mat(:, :, iFrame));
    % Write frame to video
    writeVideo(vidWriter, frame);

end
%close video stream
close(vidWriter);
```

Obrázek 17 - Ukázka kódu, který řeší zápis matice obrazů do video souboru

Příkaz *VideoWriter()* vytvoří video soubor pro zápis, ten byl následně otevřen a je prováděn zápis, důležité před samotným zápisem snímku do souboru, byl jeho převod do formátu unit8[18]. Následně pomocí funkce *close()* je ukládání dat ukončeno a připraveno pro další manipulaci.

6.2 Testování navrženého algoritmu na reálných datech

Předešlá kapitola 6.1, se zabývala návrhem algoritmu pro detekci, obrazová data použita v této kapitole, byla vytvořena pouze za účelem testování. Základní parametry těchto vytvořených dat jsou uvedeny v tabulce (Tabulka 8). V předcházející kapitole byl použit záznam číslo 3. Všechny záznamy uvedené v tabulce, byly pořízeny za svítivosti v místnosti 1000 LUX.

Tabulka 8 - Základní parametry testovaných video záznamu

č. záznamu	*rozlišení[px]*	*délka[s]*	*počet snímků*
1	240x320	32	1109
2	640x480	14	245
3	1280x720	12	300
4	1920x1080	10	300

Následující tabulka (Tabulka 9) uvádí základní parametry kamery, která byla použita pro pořízení videozáznamů uvedených výše.

[18] Unit8 – obrazový format realizovaný 0-256bity

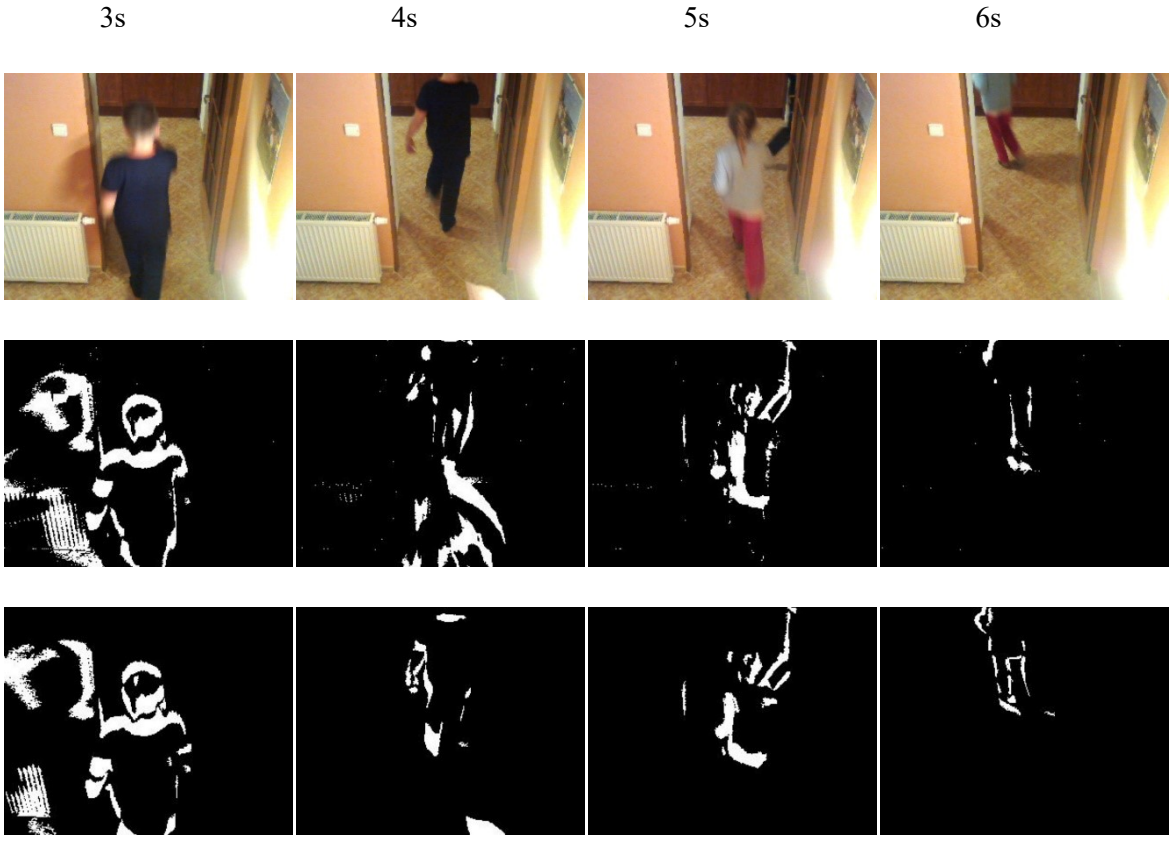
Tabulka 9 - Zakladní parametry webkamery

Název	*Hodnota*
Značka	Creative, LIVE!CAM inPerson HD
Rozlišení	1920x1080
Snímací frekvence	až 30 snímků za minutu při rozlišení 1080p (Full HD)

6.2.1 Testování video záznamu č. 1

Tento video záznam měl nejmenší rozlišení a postupně na něj byly aplikovány obě metody detekce jak je vidět na sekvenci (Sekvence obrázků 1). První řádek zobrazuje nativní záznam, druhý řádek aplikaci metody background substraction a třetí řádek sekvence použití metody frame differencing.

Bylo zde názorně vidět, že při aplikaci obou metod na video záznamy s nízkým rozlišením (240x320 px), byla detekce přesná a po užití obou metod identická. Avšak na prvním snímku v čase 3s byl vidět odlesk od topení po aplikaci obou detekcí. Na ostatních snímcích už byla detekce úspěšná bez výrazných známek zašumění obrazu. Aplikace filtru pro redukci šumu, která byla použita u metody frame differencing (viz kapitola 6.1.4), u video záznamu s nízkým rozlišením, nemá na výslednou detekci vliv.

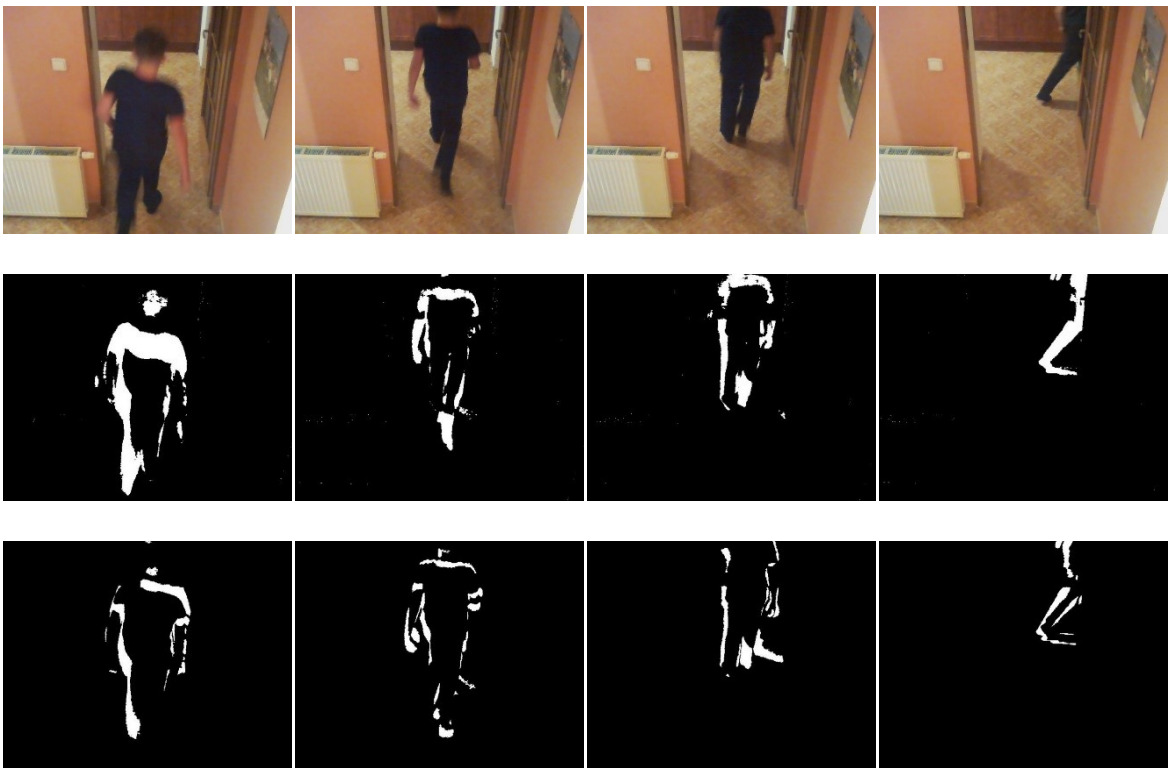


Sekvence obrázků 1 - Testování algoritmu na video záznamu č. 1, první řádek – nativní video záznam, druhý řádek – background substraction, třetí řádek – frame differencing

6.2.2 Testování video záznamu č. 2

Obrázky, které jsou uvedeny v sekvenci (Sekvence obrázků 2), znázorňují průběh testování video záznamu č. 2. Při srovnání jednotlivých metod detekce pohybu, bylo vidět, že pohyb, byl opět detekován přesně. Nicméně při detailnějším pohledu na druhý řádek sekvence, je vidět šum, který se objevuje po detekci metodou background substraction. Tento šum ale nebyl detekován při použití metody frame differencing, z důvodů aplikace filtru. Mohli jsme tedy tvrdit, že při detekci pohybu z videozáznamu s rozlišením 640x480 px, byla metoda frame differencing přesnější.

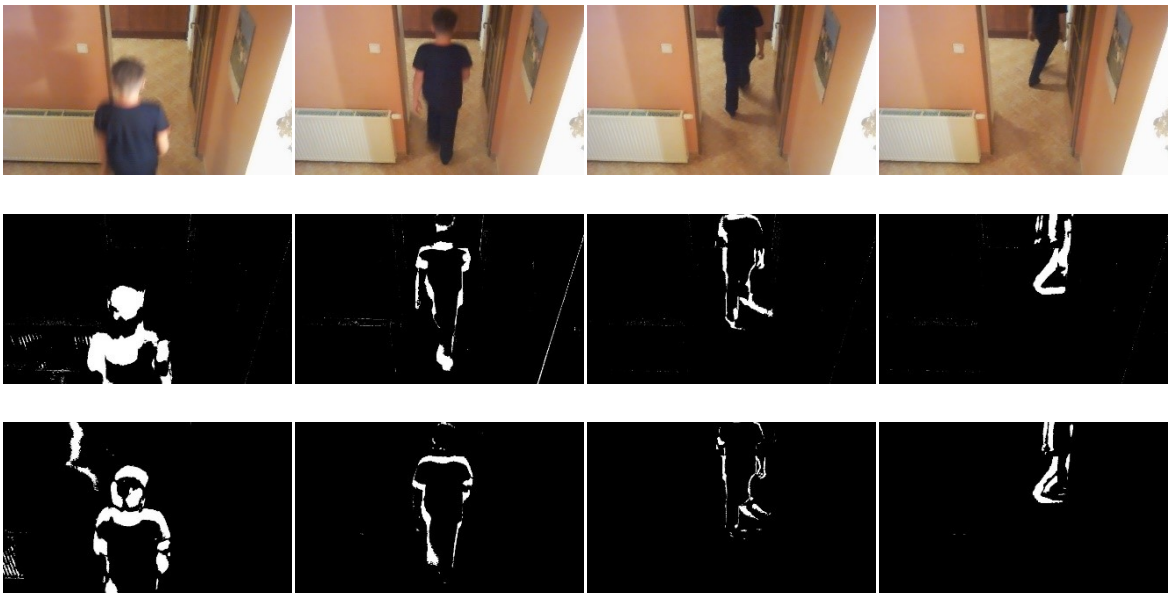
5s 5,5s 6s 7s



Sekvence obrázků 2 - Testování algoritmu na video záznamu č. 2, první řádek – nativní video záznam, druhý řádek – background substraction, třetí řádek – frame differencing

6.2.3 Testování video záznamu č. 4

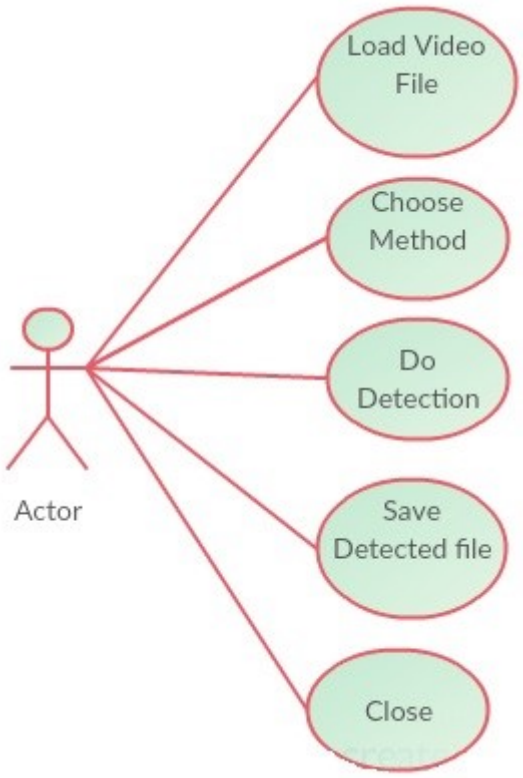
3s 3,5s 4s 4,5s



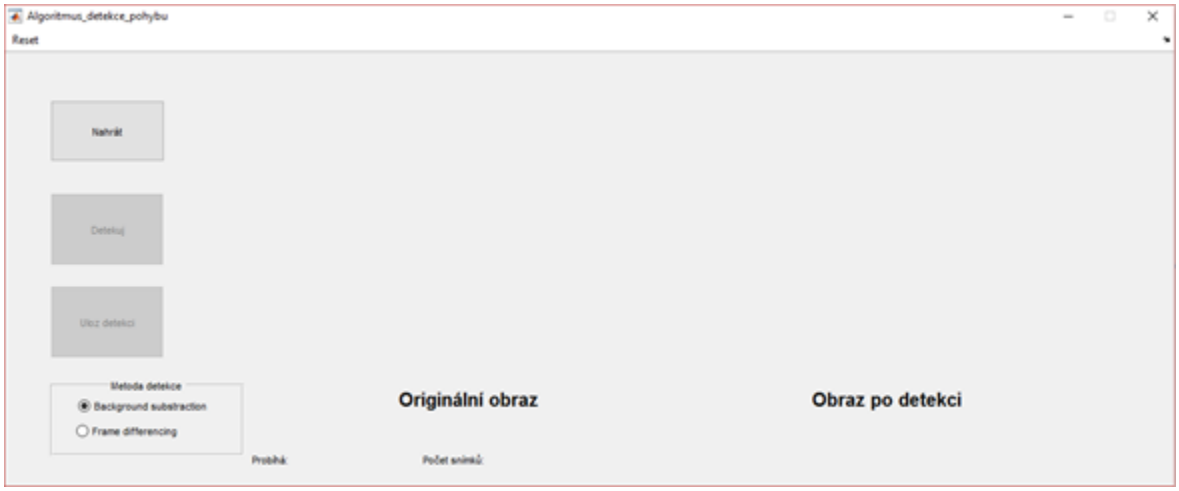
Sekvence obrázků 3 - Testování algoritmu na video záznamu č. 4, první řádek – nativní video záznam, druhý řádek – background substraction, třetí řádek – frame differencing

Poslední testovaný video záznam, byl záznam s největším rozlišením 1920x1080px, tento záznam byl nejkratší ze všech testovaných, jelikož díky velkému rozlišení je kladena velká výpočetní náročnost na hardware. Ve srovnání obou metod můžeme vidět, že frame differencing metoda (Sekvence obrázků 3, třetí řádek) je úplně bez zašuměných oblastí, oproti tomu metoda background substraction (Sekvence obrázků 3, druhý řádek, snímek pro 3s a 3,5s) vykazovala stopy šumu.

6.3 Návrh graficko-uživatelského rozhraní



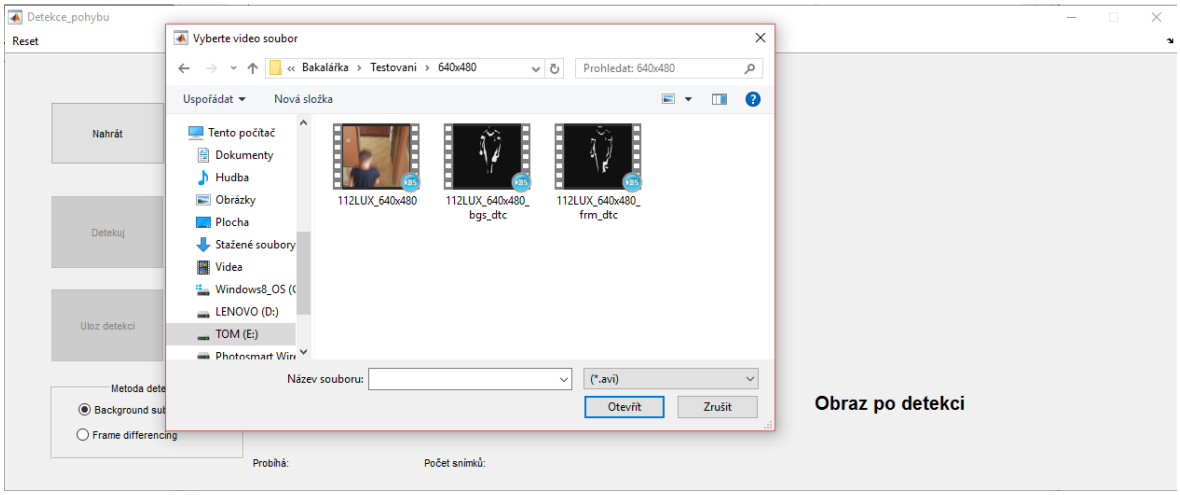
Obrázek 18 - Use case diagram navrhovaného algoritmu



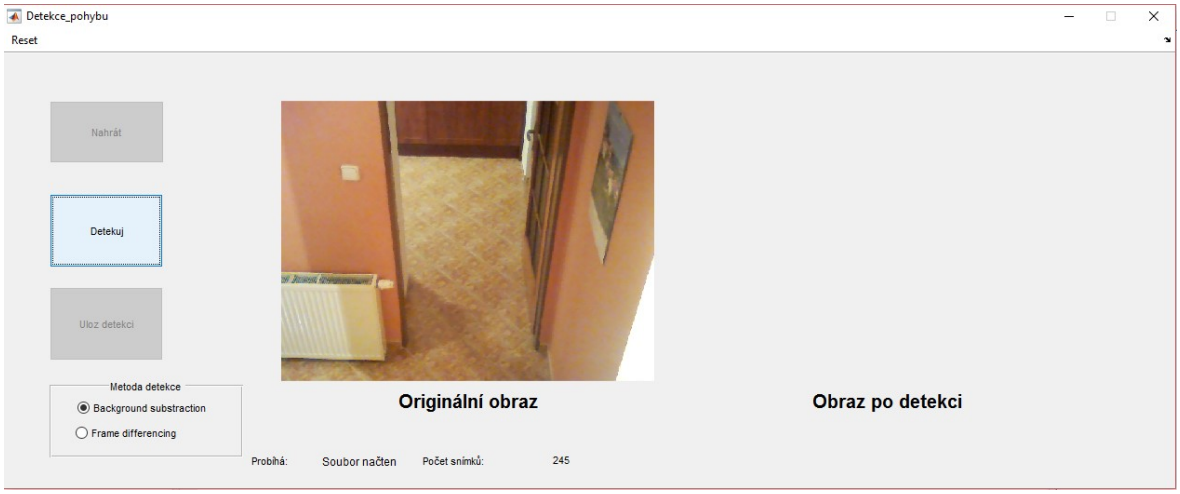
Obrázek 19 - Zobrazení GUI při spuštění programu

Obrázek 18 zobrazuje všechny možnosti, které má uživatel pracující s navrhovaným softwarem. Může tedy načíst video soubor, který musí být ve formátu avi a nesmí být delší než 30s pro záznamy s rozlišením 240x320, nebo 10s záznamy rozlišením 1280x720 a vyšším. Záznam si uživatel může vybrat po stisku tlačítka "Nahrát", jakmile je mu otevřeno dialogové okno pro výběr souboru (Obrázek 20). Jiná možnost ani uživateli není nabídnuta, jak můžeme vidět na obrázku (Obrázek 19).

Poté co je soubor vybrán, zobrazí se uživateli náhled v sekci "Originální obraz", dále je celý průběh načítání video souboru komentován v sekci "Probíhá". Náhled programu po načtení video souboru můžeme vidět na obrázku (Obrázek 21).



Obrázek 20 - Výběr video souboru, ve kterém chceme detekovat pohyb

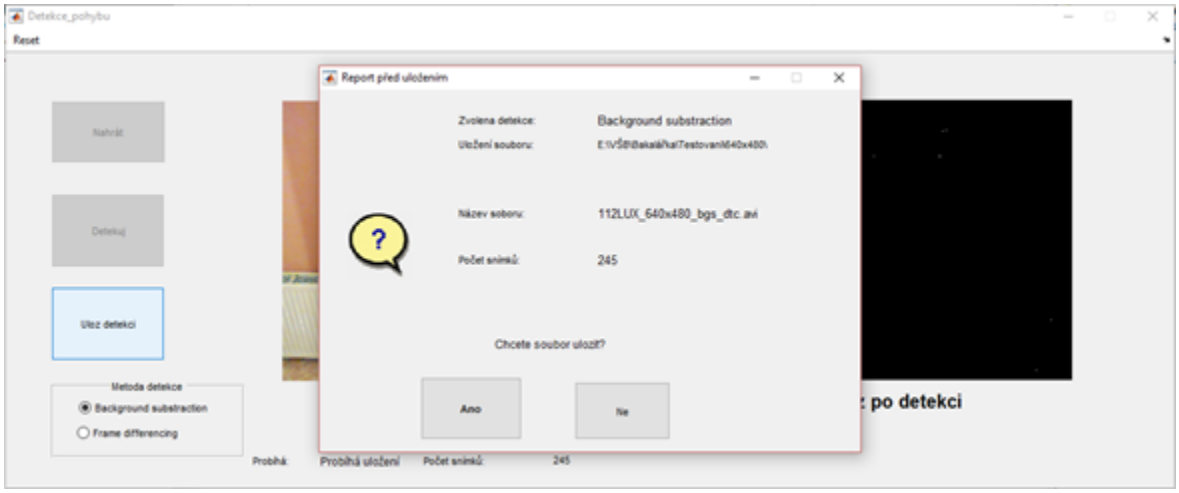


Obrázek 21 - Stav GUI po načtení video souboru

Nyní je uživateli umožněno zvolit si metodu detekce, zvolením jedné ze dvou možností detekce v sekci “Metoda detekce“, defaultně nastavena je metoda background substraction. Na náhledu z programu můžeme vidět, že uživateli je znemožněno vybrat jakoukoliv jinou možnost než “Detekuj“, pro případ, že chce uživatel zvolit jiný video záznam, slouží tlačítko “Reset“, nahoře v kontextovém menu. Možnost reset je přístupna kdykoliv v průběhu celého programu a navrátí jej do původního stavu, který je na obrázku (Obrázek 19).

Obrázek 22 - Stav rozhraní po detekci pohybu

Obrázek 22 znázorňuje stav po proběhlé detekci, to signalizuje text “Pohyb detekován“ v sekci “Probíhá“ a náhled obrazu, který se zobrazil v “Obraz po detekci“. Pokud je uživatel s provedenou detekcí spokojen klikne na tlačítko “Ulož detekci“, je mu zobrazeno dialogové okno, které shrnuje základní parametry celé detekce, jak můžeme vidět na obrázku (Obrázek 23). Po stisknutí tlačítka “Ano“, je program navrácen do původního stavu a připraven pro novou detekci pohybu. Při stisknutí možnosti “Ne“ je program resetován do původního stavu (Obrázek 19).



Obrázek 23 - Zobrazení dialogu před uložením záznamu detekce

7 Závěr

Úkolem bakalářské práce bylo vytvořit algoritmus pro detekci pohybu z video záznamu. Za tímto úkolem byla v první části vypracována komplexní rešerše týkající se problematiky detekce pohybu a dané problematiky. Na základě získaných poznatků byly zvoleny dvě metody (background substraction a frame differencing) a vytvořen vhodný algoritmus, který je schopen detekovat pohyb. Algoritmus obsahuje všechny důležité fáze zpracování videa od jeho načtení, přes segmentaci, samotnou detekci až po závěrečné uložení. Celý navržený algoritmus byl dále testován na experimentálně vytvořených datech. Jednalo se o video záznamy pořízené webkamerou Creative. Testovací video záznamy byly pořízeny v chodbě bytu za konstantního osvětlení 1000 LUX s rozlišením 240x320px, 640x480px, 1280x720xp a 1920x1080px. Nejlepších výsledků dosáhl algoritmus při rozlišení 1280x720px nebo nižším a použití metody frame differencing. Při použití metody background substraction se ve výsledných záznamech obsahujících detekci objevovalo velké zašumění způsobené odlesky od okolních povrchů (topení, rámy dveří). V rámci bakalářské práce bylo vytvořeno intuitivní graficko-uživatelské rozhraní, tak aby běžný uživatel mohl s programem pracovat co nejrychleji a nejefektněji. Graficko-uživatelské rozhraní obsahuje možnost volby metody, kterou chceme detekovat a taky možnost náhledu nativního video záznamu a záznamu po detekci. Hlavní nevýhodou celého navrhovaného algoritmu je, že není schopen detekovat pohyb v reálném čase. Nejprve tedy musíme pořídit záznam, před tím než provedeme samotnou detekci. Toto je pro aplikaci např. v home care systémech pro sledování seniorů a pozdější detekci pádu nežádoucí. Tato práce tak dále slouží jako edukativní text pro rozšíření znalostí v oblasti zpracování dynamického obrazu v programu MATLAB. Pro tvorbu softwaru, který by fungoval v reálném čase, je však potřeba hlubších znalostí obrazové analýzy a práce s programovacím rozhraním MATLAB, zcela přesahujících rámec této bakalářské práce.

Použitá literatura

[1] KŘÍŽ, Karel. *Detekce dopravních prostředků a vyhodnocování jejich stavů z videozáznamů křižovatek* [online]. Brno, 2013 [cit. 2016-04-20]. Dostupné z: http://is.muni.cz/th/395613/fi_m/xkriz8_DP.pdf?zpet=https:%2F%2Ftheses.cz%2Fvyhledavani%2F%3Fsearch%3DDetekce%20pohybu%20z%20videoz%C3%A1znamu%26start%3D1. Diplomová práce. Masarykova Univerzita Brno - Fakulta Informatiky. Vedoucí práce RNDr. David Svoboda, Ph.D.

[2] TEKALP, A. Digital video processing. Second edition. New York: Prentice Hall, 2015. ISBN 01-339-9100-8.

[3] Handbook of image and video processing. Editor Alan C BOVIK. San Diego: Academic Press, c2000. ISBN 0121197905.

[4] Handbook of image and video processing. 2nd ed. Editor Alan C BOVIK. Burlington: Elsevier/Academic Press, c2005. ISBN 0121197921.

[5] CASTLEMAN, Kenneth R. Digital image processing. Upper Saddle River: Prentice Hall, c1996. ISBN 0132114674.

[6] RUSS, John C. The image processing handbook. 5th ed. Boca Raton: CRC/Taylor and Francis, c2007. ISBN 978-084-9372-544.

[7] HLAVÁČ, Václav. Digitální obraz, základní pojmy. In: Fakulta elektrotechnická, katedra kybernetiky [online]. Praha, 2010 [cit. 2016-04-20]. Dostupné z: http://cmp.felk.cvut.cz/~hlavac/TeachPresCz/11DigZprObr/014DigitalImageCz.pdf

[8] GONZALEZ, Rafael C a Richard E WOODS. Digital image processing. 3rd ed. Upper Saddle River: Pearson, c2008. ISBN 01-316-8728-X.

[9] Základní parametry digitálního obrazu. Www.senpai.cz [online]. 2014 [cit. 2016-04-20]. Dostupné z: http://it.senpai.cz/7-lekce-zakladni-parametry-digitalniho-obrazu

[10] BÁBÍČEK, Radek. Digitální video na počítači. Vyd. 1. Brno: Computer Press, 2006. Jak na počítač (Computer Press). ISBN 80-251-0830-9.

[11] MICHALIK, Pavel. Digitální video v praxi - technické základy: učební text pro předmět U068. Vyd. 1. Praha [i.e. Brno]: Tribun EU, 2007. Knihovnicka.cz. ISBN 978-80-7399-220-0.

[12] KUNDUR, DR., Deepa. Introduction to Video Processing. University of Toronto.

[13] PICCARDI, M. Background subtraction techniques: a review. 2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No.04CH37583). IEEE, 2004, , 3099-3104. DOI: 10.1109/ICSMC.2004.1400815. ISBN 0-7803-8567-5. Dostupné také z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1400815

[14] GONZALEZ, Rafael C, Richard E WOODS a Steven L EDDINS. Digital image processing using MATLAB. Upper Saddle River: Pearson Prentice Hall, c2004. ISBN 0-13-008519-7.

[15] REYES-ALDASORO, Constantino Carlos. Biomedical image analysis recipes in MATLAB: for life scientists and engineers. Hoboken, NJ: American Geophysical Union, 2015. ISBN 978-111-8657-553.

[16] MARQUES, Oge. Practical image and video processing using MATLAB [online]. Hoboken, N.J.: Wiley-IEEE Press, 2011 [cit. 2016-04-24]. ISBN 9781118093481.

[17 BLANCHET, Gérard a Maurice CHARBIT. Digital signal and image processing using MATLAB. Přeložil Antoine HERVIER. London: ISTE, 2006. ISBN 1-905209-13-4.

[18] MATLAB Documentation [online]. US: MathWorks, 2016 [cit. 2016-04-24]. Dostupné z: http://www.mathworks.com/help/matlab/?refresh=true

[19] HSU, Chen-Chien, Wei-Chen LIU a Chin-Ming HONG. Real-time recognition of moving direction of a target object based on historical trajectories. Proceedings 2011 International Conference on System Science and Engineering. IEEE, 2011, , 70-75. DOI: 10.1109/ICSSE.2011.5961876. ISBN 978-1-61284-351-3. Dostupné také z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5961876

[20] NEMADE, Bhushan a Vinayak Ashok BHARADI. Adaptive automatic tracking, learning and detection of any real time object in the video stream. In: 2014 5th International Conference - Confluence The Next Generation Information Technology Summit (Confluence) [online]. 2014 [cit. 2015-06-02]. DOI: 10.1109/confluence.2014.6949039.

[21] ABDELALI, Hamd Ait, Fedwa ESSANNOUNI, Leila ESSANNOUNI a Driss ABOUTAJDINE. Algorithm for moving object detection and tracking in video sequence using color feature. In: 2014 Second World Conference on Complex Systems (WCCS) [online]. 2014 [cit. 2015-06-02]. DOI: 10.1109/icocs.2014.7060999.

[22] YANG, Mingyang. A moving objects detection algorithm in video sequence. In:2014 International Conference on Audio, Language and Image Processing[online]. 2014 [cit. 2015-06-02]. DOI: 10.1109/icalip.2014.7009826.

[23] LIANG ZHANG,, XIANGMING WEN, WEI ZHENG a BO WANG. An algorithm for moving semantic objects trajectories detection in video. 2010 IEEE International Conference on Information Theory and Information Security. IEEE, 2010, : 34-37. DOI: 10.1109/ICITIS.2010.5689631. ISBN 978-1-4244-6942-0. Dostupné také z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5689631

Seznam příloh

Příloha I Obsah disku DVD

- **Elektronická verze bakalářské práce**
- **Soubor *Detekce_pohybu_GUI.m* (spustitelný pouze v programu MATLAB)**
- **Soubor *rprtBox.fig* a *rprtBox.m* (spustitelný pouze v programu MATLAB)**